ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE
Laboratoire Européen pour la Physique des Particules

CERN - PS DIVISION

CTF / Note 98-07

LOGICIEL D'IMAGERIE LASER - BEAMDIAG.

Stéphan DEL BURGO

Résumé

Ce rapport présente le fonctionnement, l'architecture et les fonctions du logiciel BEAMDIAG utilisé pour l'imagerie laser dans les laboratoires Laser CTF (2013-1-403), Photocathodes et Lasers (101-R-010), et Laser (169-S-043) du groupe PS/LP.

Cette note décrit la dernière version (12/97) du programme BEAMDIAG, celle développée pour l'environnement de Matlab 5. Historiquement, il s'agit de la deuxième version de ce programme, qui existait déjà pour l'environnement de Matlab 4. Cette version reprend l'architecture générale du programme précédent (12/96) avec quelques modifications imposées par Matlab 5.

Cette note a pour but, outre celui d'expliquer l'utilisation du logiciel, de permettre au programmeur de modifier le programme pour des personnalisations et des améliorations.

Le listing du programme est disponible en annexe à la fin du document.

Sommaire - Logiciel d'imagerie laser - BEAMDIAG

LOGICIEL D'IMAGERIE LASER - BEAMDIAG.

I. Introduction :

Une bonne connaissance de la géométrie des faisceaux lasers est indispensable pour améliorer les réglages, et ainsi le fonctionnement général et les performances du laser. Si les radiations utilisées ne sont pas dans le spectre visible, si les impulsions lasers sont très courtes, quelques picosecondes ou quelques nanosecondes, les réglages deviennent délicats.

Un système d'imagerie numérique a été développé pour la visualisation de la géométrie des faisceaux lasers. Il fonctionne régulièrement dans les laboratoires Laser CTF (2013-1-403), Photocathodes et Lasers (101-R-010), et Laser (169-S-043) du groupe PS/LP.

La partie matérielle du système d'imagerie ne sera pas décrite dans cette note. Mais comme elle a une influence sur le mode de fonctionnement du logiciel, elle sera succinctement présentée.

I.1. Matériel :

Le faisceau à analyser est observé par une caméra. Suivant la longueur d'onde analysée, un écran scintillant peut être utilisé pour rendre le faisceau visible pour la matrice CCD de la caméra. Un écran en CsI(Tl) est, par exemple, utilisé pour visualiser les faisceaux UV (262 nm) des lasers du CTF.

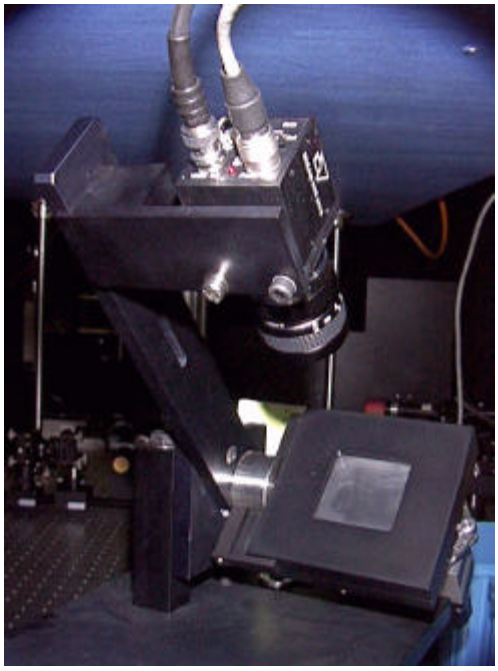

fig 1 : Caméra et écran scintillant.

Les impulsions lasers de quelques picosecondes ou quelques nanosecondes, avec une fréquence de répétition de 10 Hz, nécessitent un système de timing précis pour être correctement visualisées par la caméra. Sachant qu'une trame de l'image dure 20 ms (deux trames entrelacées

pour une image vidéo classique à 25 Hz), l'impulsion laser n'est présente que sur une trame, l'autre étant vide. Le système de timing, mis au point par M. Eric Chevallay, permet de synchroniser parfaitement les acquisitions avec les impulsions laser. Chaque impulsion apparaît sur une trame précise (paire ou impaire suivant la caméra) de la vidéo.

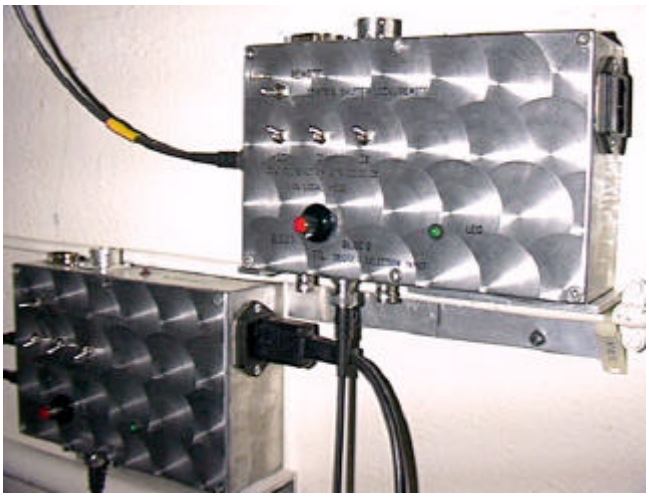

fig 2 : Système de timing.

Les données de l'image sont ensuite récupérées par le frame grabber installé dans le PC. A l'aide du frame grabber, plusieurs opérations sont effectuées. Une région d'intérêt de la matrice CCD est sélectionnée. Cela permet de minimiser la taille des fichiers traités, en fonction de la partie utile de l'image. Seuls les pixels de la trame comprenant l'impulsion laser sont conservés. Ainsi, une ligne sur deux de la matrice CCD apparaît dans le fichier image (résolution inférieure en vertical). Enfin, le bruit sur l'image, mesuré sur une image sans faisceau, est soustrait à l'image entière.

A la sortie du frame grabber, le résultat est un fichier TIFF que l'on lit sur le disque dur (fichier **'c:\video\images\image.tif'**). C'est une matrice de taille variable suivant la région sélectionnée du CCD et la position de la caméra.

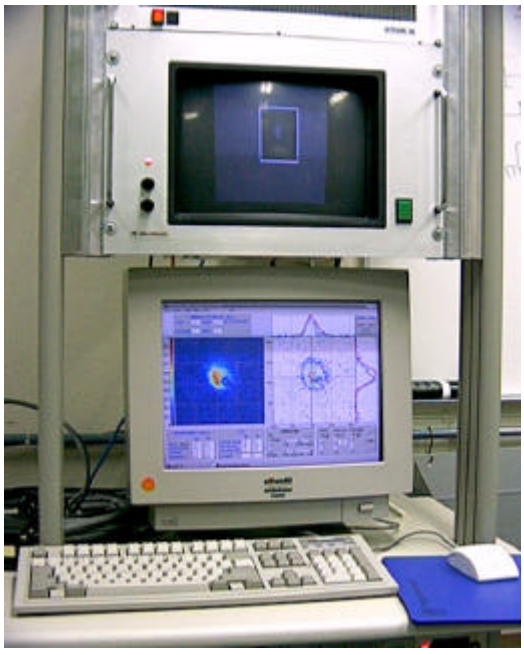

fig 3 : PC utilisé pour l'imagerie.

I.2. Logiciel :

La taille de la matrice de l'image peut être très grande (par exemple 256x512). Matlab permet de manipuler facilement de grandes matrices et de les afficher comme des images. Le logiciel d'imagerie est programmé avec Matlab.

La première version du logiciel (12/96) utilisait Matlab 4. L'impression et d'autres petits détails étaient alors mal gérés par Matlab 4. Maintenant, ces problèmes sont résolus avec Matlab 5. Matlab 5 est également un peu plus rapide que son prédécesseur, pour l'affichage des graphiques, mais cela nécessite une machine rapide et performante (PC 200 MHz par exemple).

Le programme comprend 77 fichiers, dans cette version. Il serait long et fastidieux de tous les décrire. Le fichier principal s'appelle **'beamctf'** et est constitué de plusieurs parties. Nous l'étudierons en détail dans le prochain paragraphe. On trouvera le listing du programme en annexe.

II. Architecture :

Nous allons décrire l'architecture afin de comprendre l'organisation générale du programme.

II.1. Programme principal - démarrage :

L'ensemble des fichiers du programme **BEAMDIAG** se trouve dans un répertoire dont le nom dépend de la machine sur laquelle il est installé (par exemple **'matlab\ctf'**). Dans ce répertoire se trouve le programme principal **'beamctf'**, qui est divisé en différents sous-programmes.

Quatre sous-programmes peuvent être appelés dans **'beamctf'** :

- **'InitializeBEAM'** pour initialiser les paramètres et dessiner les figures où apparaîtront les images du faisceau à l'écran,
- **'DisplayBEAM'** pour faire l'analyse du faisceau proprement dite,
- **'SaveBEAM'** pour sauvegarder dans un fichier l'image à l'écran, ainsi que les données de l'image,
- **'CloseBEAM'** pour libérer la mémoire et quitter le programme.

Il existe différentes façons de lancer le programme. A partir de Matlab 5 directement, il suffit d'exécuter l'instruction **beamctf('InitializeBEAM',numcam,pxlx,pxly)**. On peut également lancer le programme à partir de Windows 95, en exécutant un programme qui appelle Matlab et exécute l'instruction **beamctf('InitializeBEAM',numcam,pxlx,pxly)** ou mieux à partir d'un 'shortcut' qui appelle ce programme (cf. les programmes **'dblaseruv.wi2'** et **'dblaseruv.m'**).

Le paramètre **numcam** correspond au numéro de la caméra que l'on désire utiliser, si l'on travaille avec un faisceau, ou au numéro du mode de calcul de l'image simulée si l'on travaille en mode simulation (sans faisceau).

Les paramètres **pxlx** et **pxly** sont les dimensions en millimètres que l'on affecte aux pixels de l'image (dimensions d'un pixel respectivement en horizontal (x) et en vertical (y)) pour avoir

une échelle à l'écran correspondant aux dimensions réelles. Ces dimensions sont utilisées pour avoir à l'écran une fenêtre d'affichage orthonormée. La région d'intérêt sur le CCD est calculée pour avoir, à l'affichage, une image qui tient compte des éventuelles projections dues à l'écran scintillant ou autre élément optique situé entre le point d'observation du faisceau et la caméra.

Si le faisceau à observer est circulaire, il est également circulaire à l'écran si les paramètres **pxlx** et **pxly** sont corrects, c'est à dire s'ils tiennent compte des projections éventuelles (et du grossissement de l'optique s'il y en a). Pour le paramètre **pxly** (pixel en y), il ne faut pas tenir compte du fait qu'une seule trame est utilisée, et qu'à l'affichage, la trame vide n'apparaît pas. La valeur de **pxly** correspond donc, au grossissement près, à la dimension physique du pixel de la CCD en y (soit (pixel + interpixel) $\times$ projection $\times$ grossissement). La correction due à la trame manquante est apportée dans la variable globale **PXLY**.

II.2. 'InitializeBEAM' :

Le sous-progamme **'InitializeBEAM'** définit tous les paramètres et prépare l'écran pour l'affichage.

Les variables globales (noms en majuscules sauf **camera**), utilisées par les autres programmes ou sous-programmes, sont déclarées. Elles sont, de la même manière, supprimées dans **'CloseBEAM'**.

La fenêtre principale d'affichage est définie. Son nom ('Tag') est **'BEAMDIAG'** et ses références ('Handle') sont sauvegardées dans la variable **PARA(1)**. Le vecteur **PARA** contient toutes les références des objets (graphiques, indicateurs, etc.) créés par Matlab et permet de rappeler ces objets à partir de ses références. La liste des objets sauvegardés dans **PARA** se trouve plus loin dans la note.

Ensuite, le menu de la fenêtre est défini avec la fonction **'menubeam'**. Les paramètres par défaut pour les différentes options d'analyse sont sauvegardés dans la variable **BEAM_DAT**.

Puis, les paramètres pour les dimensions de la région d'intérêt sur la matrice CCD sont calculés en fonction de la diagonale de l'image. Les fonctions **'width'** et **'height'** calculent les dimensions de l'image en pixels pour que celle-ci soit orthonormée.

Ces calculs ne sont effectués que si les acquisitions proviennent d'une caméra (cf. la variable **PROG**). En effet, il est possible de faire des analyses avec des images de faisceau calculées ou avec des images d'archives. Dans ces cas, aucune région d'intérêt n'est définie.

Enfin, différents paramètres sont définis :

- **COLUMN** : nombre de colonnes de l'image,
- **ROW** : nombre de lignes de l'image,
- **camera** : matrice de l'image provenant de la caméra ou calculée,
- **PREVIOUS** : position précédente du barycentre ou du maximum de l'image,
- **REF** : point de référence dans l'image,
- **PERSONAL** : point défini par l'utilisateur pour l'analyse des profils,
- **BARTH** : seuils bas et haut pour le calcul du barycentre,
- **EPSILON** : différence minimale entre les seuils du barycentre,
- **CONTOURTH** : seuils des différents contours.

La fonction **'graduate'** prépare les graduations (variables **ABSCISSA** et **ORDINATE**) en millimètres pour les axes des graphiques. La valeur des graduations est calculée pour qu'elles coïncident avec des valeurs rondes et qu'il y ait au moins quatre graduations sur l'axe.

Les différents cadres pour les affichages et les commandes (boutons, ascenseurs,…) sont dessinés avec la fonction **'setframe'**. Les commandes pour le contrôle de la région d'intérêt de la matrice CCD sont définies par la fonction **'initwind'**, les axes pour l'affichage des images par **'initaxes'** avec **'newview'** et **'newcont'**.

Enfin, l'écran est mis à jour. Les figures apparaissent.

II.3. La variable PROG :

La variable **PROG**, qui est définie en début de programme (**'beamctf'**), peut prendre quatre valeurs :

- 1 : les images proviennent de la caméra et sont analysées en continu (boucle),
- 2 : les images proviennent de la caméra et sont analysées image par image (à chaque pression sur le bouton **PLAY**),
- 3 : les images proviennent de fichiers spécifiques (par exemple du fichier **'j:\data\cern\image\fichier.tiff'** et sont lues en boucle,
- 4 : les images sont calculées selon un modèle.

Dans le mode 4 (simulation), différents modes de calcul sont possibles suivant le paramètre **numcam** donné lors du lancement du programme :

- 1 : mode gaussien TEM 00,
- 2 : mode gaussien TEM 00 avec ajout de bruit gaussien,
- 3 : mode gaussien TEM 10,
- 4 : mode gaussien TEM 10 avec ajout de bruit gaussien,
- 5 : mode gaussien TEM 01,
- 6 : mode gaussien TEM 01 avec ajout de bruit gaussien,
- 7 : mode gaussien TEM 11,
- 8 : mode gaussien TEM 11 avec ajout de bruit gaussien.

Ces calculs sont effectués par la fonction **'simbeam'**. Ce mode d'analyse est très pratique pour la mise au point du programme.

II.4. La boucle principale 'DisplayBEAM':

L'analyse du faisceau se fait dans le sous-programme **'DisplayBEAM'** de **'beamctf'**. Il faut appuyer sur le bouton **PLAY** pour lancer son exécution. Pour l'arrêter, il faut appuyer sur le bouton **STOP** (sauf si **PROG=2**).

Avant d'entrer dans la boucle, les paramètres sont initialisés. Les références des graphiques sont rappelées pour pouvoir les commander directement. Les boutons sont désactivés ou activés suivant les cas.

Le paramètre **'Userdata'** de **PARA(4)** (bouton **PLAY**) est mis à **1**. La boucle est ainsi décrite tant qu'il a cette valeur. Le fait d'appuyer sur **STOP** remet ce paramètre à 0.

Aquisition :

La première action réalisée dans la boucle est l'acquisition de l'image. Dans le cas des acquisitions avec la caméra (**PROG=1** ou **2**), différentes routines sont appelées suivant la caméra utilisées (cf. **acqvideo** ou **acqdrive** et **oddtra~1.pif** ou **eventr~1.pif**). L'image est ensuite récupérée dans Matlab au format TIFF et convertie en matrice d'intensité. La martice s'appelle alors **camera** (variable globale). C'est l'image provenant directement de la vidéo (pas de normalisation, pas de filtrage, seulement une conversion de code).

Dans cette matrice intensité, le niveau des pixels est codé entre 0 et 1 de façon linéaire (0 : pas de signal , 1 : saturation). C'est plus facile à manipuler que le code TIFF qui va de 0 à 255 avec la correspondance RGB.

Si le mode simulation est utilisé, la matrice **camera** est directement calculée avec des valeurs comprises entre 0 et 1 suivant le modèle choisi. Si les images sont des images d'archives, elles sont chargées et converties, si besoin est, pour obtenir la matrice **camera**.

Une fois l'image acquise, l'analyse commence.

Niveaux et saturation :

Les niveaux vidéo de l'image sont testés. Si la plage de niveaux utilisée est trop étroite (**(HighLevel-LowLevel)<LOOPLVL**), les analyses ne sont pas effectuées et on fait une nouvelle acquisition.

D'autre part, un signal de **SATURATION** (paragraphe ***'saturation'*** plus loin dans la boucle) apparaît à l'écran si l'image comprend beaucoup de pixels au niveau maximal (définition du seuil : 20 pixels dont le niveau est supérieur à 0,95). Dans ce cas, les analyses ne sont pas correctes et ce mode de fonctionnement est dangereux pour la matrice CCD (réduction de la durée de vie). Il faut alors atténuer le faisceau arrivant sur la caméra.

Le test sur la saturation du CCD est adapté pour les analyses avec les faisceaux UV et IR. Il convient d'être attentif au fonctionnement de la caméra avec les autres longueurs d'onde (en particulier avec le vert (532 nm)). La saturation peut se produire à des niveaux inférieurs (0,8 au lieu de 1). Dans ce cas, le signal de saturation n'apparaît pas (si le seuil n'est pas modifié), bien que l'analyse soit incorrecte.

Normalisation :

Pour utiliser toute la dynamique disponible pour l'affichage des images, l'image est normalisée. La normalisation consiste en une transformation linéaire de l'histogramme de l'image, afin que le plus petit niveau de l'image soit 0 et le niveau maximal soit 1. On obtient une nouvelle matrice que l'on appelle **normalized**.

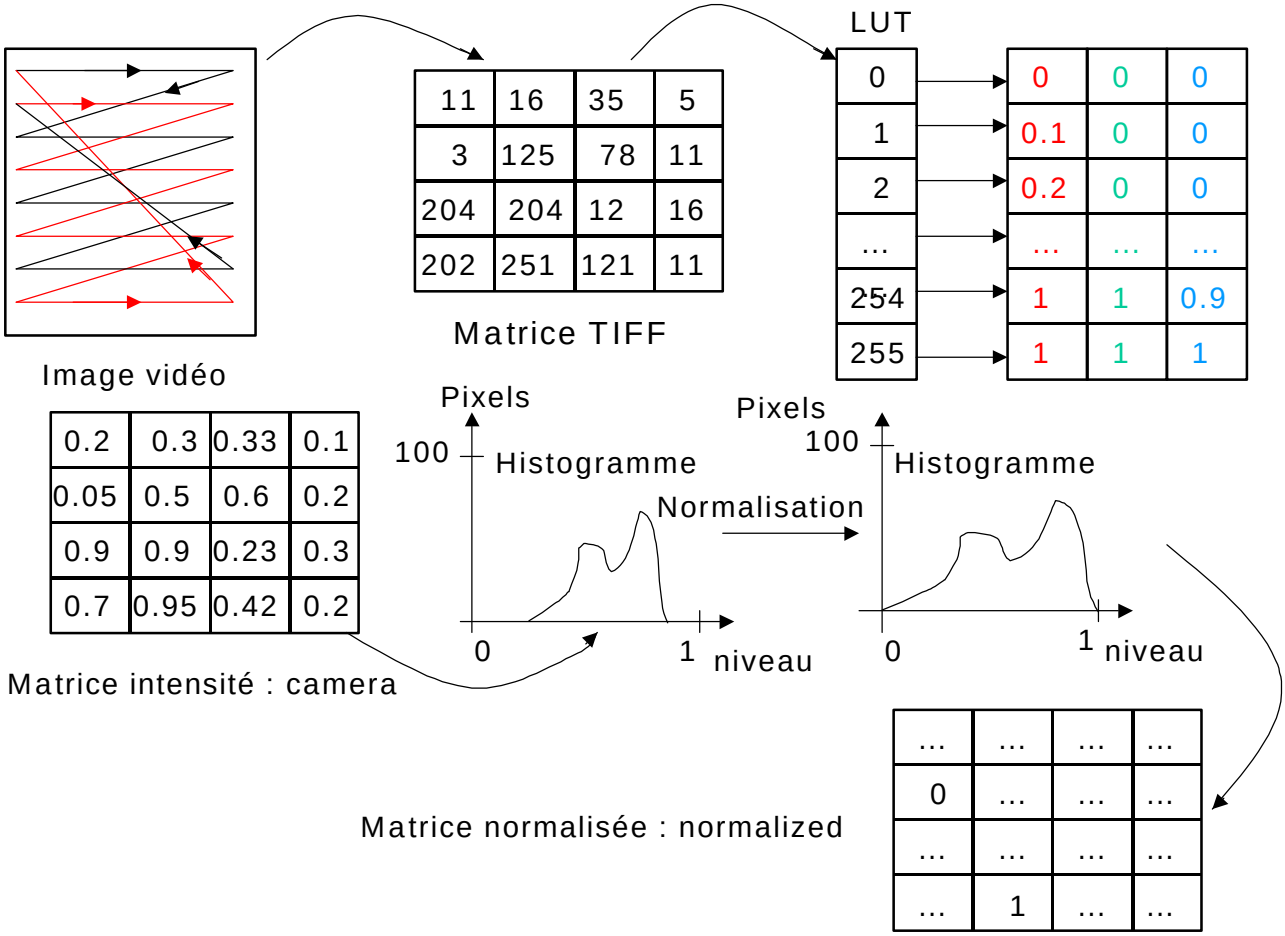

fig 4 : Conversion et normalisation.

Ecran :

L'écran est divisé en différentes parties d'affichage. A gauche (**ViewAxes**), est présentée l'image sans calcul avec un code de couleur particulier (colormap jet). A droite (**AnalysisAxes**), sont présentés les résultats de l'analyse avec le tracé des contours, le tracé des profils ou le tracé en trois dimensions.

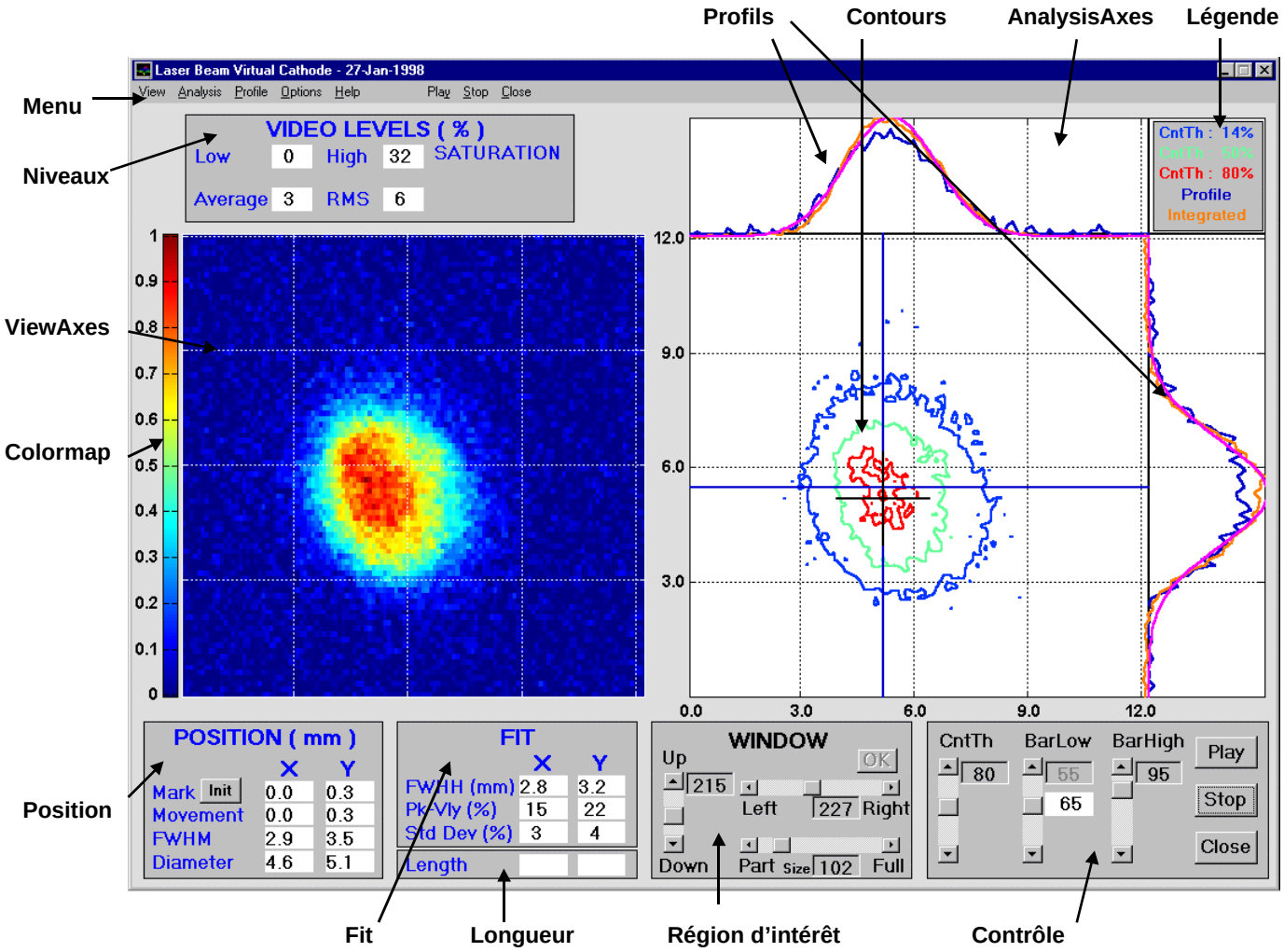

fig 5 : Ecran d'affichage.

ViewAxes :

Dans **ViewAxes**, soit la matrice **camera**, soit la matrice **normalized** est affichée. Le choix se fait à partir du menu **View**.

Les calculs suivants ne concernent plus que les graphiques de l'**AnalysisView** et les résultats de **POSITION** et **FIT**.

Bruit :

Par défaut, le bruit de fond (fonction **'noise'**) de l'image est soustrait à la matrice **normalized**, mais ceci peut être supprimé à partir du menu **Options** avec **Background**. La matrice **normalized** est alors modifiée. Elle ne correspond plus exactement à la matrice **camera** normalisée.

Barycentre :

Le calcul du barycentre est fait en ne considérant que les pixels dont le niveau est compris dans une fourchette dont le seuil bas est choisi en fonction des niveaux de l'image (**Automatic** dans le menu **Options**) ou par l'utilisateur (**Barlow** et **Manual** dans **Options**). Le

seuil bas est affiché à chaque fois dans la fenêtre de contrôle sous **Barlow**. Le seuil haut peut être défini par l'utilisateur en modifiant **BarHigh** dans la fenêtre de contrôle.

Maximum :

La position du maximum de l'image est également calculée. Il s'agit, plus exactement, du point médian de l'ensemble des points de la matrice **normalized** dont le niveau est supérieur à 0,92 (modifiable par le programmeur).

A partir d'ici commence l'affichage dans **AnalysisView**.

3D :

Si le mode **3D** est choisi (dans le menu **Analysis**), les axes pour cet affichage sont préparés. Dans ce cas, l'image **normalized** est affichée en 3D après un réduction du nombre de points de l'image pour accélérer l'affichage. L'image en 3d peut être dessinée à l'aide de lignes (**BEAM_DAT(12)=0**) ou à l'aide de facettes (**BEAM_DAT(12)=1**).

Une croix bleue est également affichée (si choisie) et indique la position du barycentre ou du maximum de l'image (choix dans le menu **Analysis**).

Contours :

Dans le mode **Contour** (menu **Analysis**), l'image **normalized** est analysée et trois courbes de niveau sont affichées : en bleu, le niveau situé à 14% ($1/e^2$) du maximum (1), en vert, le niveau à 50 % et en rouge, le niveau indiqué par **CntTh** (choix entre 55% et 99% par l'utilisateur, 80% par défaut). Les couleurs des courbes de niveau sont définies par **'DefaultAxesColorOrder'** qui est une propriété de la figure **'BEAMDIAG'** (définition dans la partie **'InitializeBEAM'**). La croix bleue (barycentre ou maximum) peut également être affichée.

Profils :

Les profils sont ensuite définis. A partir du menu **Profile**, différents profils peuvent être choisis. En bleu peut être affiché le profil selon deux axes dont le point d'intersection est le barycentre, le maximum ou un point particulier choisi dans l'image (**AnalysisView**) à l'aide de la souris. Le profil intégré (en orange) suivant les deux directions (x et y) peut être calculé . En rose peut apparaître les profils résultats des fits.

Les profils bleus (sections suivant deux axes) ne sont pas normalisés. Ils correspondent aux valeurs de la matrice **normalized**. Les profils intégrés sont, par contre, normalisés pour que le maximum corresponde au maximum de l'affichage. La fonction **'backgrnd'** retire l'offset de ces profils. C'est pour cela qu'il peut y avoir une partie des profils qui se trouve sous l'axe (partie négative). Les calculs de largeur de profil effectués sur les profils intégrés (choix dans le menu **Options**) sont alors un peu plus réalistes. En effet, l'intégration du bruit sur toute la largeur de l'image peut créer un offset conséquent qui fausse, s'il n'est pas corrigé, les mesures de largeur.

Largeur :

Voici comment sont effectuées les mesures de largeur figurant dans la fenêtre **POSITION**. Les mesures sont faites à partir des profils, soit intégrés, soit les sections (profils bleus) suivant l'option choisie dans le menu **Options**. Les profils pris en compte, dans le cas du choix des sections, sont les profils choisis dans le menu **Profile**, ou s'ils sont désactivés les profils

suivant les axes de la croix bleue, ou ayant pour point de référence le barycentre si la croix est désactivée.

Dans tous les cas, les calculs sont effectués de la même manière. De part et d'autre du point maximum le long du profil, les premiers points se situant au niveau recherché (50% pour **FWHM** et 14% pour **Diameter**) sont pris en compte. La distance en pixels entre ces points est convertit en millimètres à l'aide des fonctions **'scaleX'** et **'scaleY'**. S'il n'y a pas de solutions, une chaîne vide est renvoyée.

Fit :

Le calcul du **FIT** n'est effectué que si l'option est choisie dans le menu **Options**. Le **FIT** est réalisé suivant les profils ou les profils intégrés. Si le **FIT** est fait selon les profils, ce sont les profils en bleu qui sont choisis, ou les profils selon la croix bleue si les profils bleus ne sont pas sélectionnés, ou les profils ayant pour point de référence le barycentre si la croix bleue n'est pas sélectionnée.

Le calcul fait appel à la fonction **'fitting'**. La fonction **'fitting'** calcule les paramètres initiaux (amplitude, offset, centre, largeur) pour démarrer le **FIT** avec la routine **'nlinfit'** selon une fonction modèle (ici une gaussienne définie dans **'gausbeam'**).

Le profil du **FIT** est tracé en rose avec les autres profils. Dans la fenêtre **FIT**, sont indiqués la largeur à mi-hauteur (**FWHM**) des courbes obtenues ainsi que l'erreur par rapport aux profils modèles. L'écart maximal ((max-min)×100) entre le profil et le fit est donné par **PK-Vly** (peak to valley) et la déviation standard de cet écart (×100) par **Std Dev**.

Tous les calculs sont terminés. Les graphiques sont mis à jours et les résultats sont affichés en dernier. Voici ceux dont nous n'avons pas encore parlés.

Autres résultats :

Dans la fenêtre **POSITION**, **Mark** indique en millimètres la position de la croix bleue ou du barycentre. Le point de référence est l'origine des axes ou la croix noire (**Mark Reference** dans le menu **Options**). A l'aide du bouton **Init**, la référence peut être initialisée à la position de la croix bleue, ou au barycentre, ou à un point particulier si la croix bleue est désactivée et l'option **personal** choisie dans **Profile**.

Movement indique le déplacement en millimètres de la croix bleue ou du barycentre par rapport à l'image précédente.

Dans la fenêtre **VIDEO LEVELS**, **Low** et **High** indiquent en pourcentage les valeurs extrêmes de la matrice intensité **camera** (pas de normalisation, pas de filtrage, 1 (saturation) donne 100%), **Average** et **RMS**, la moyenne et la déviation standard de cette matrice.

Dans la fenêtre **Length**, se trouve la distance en millimètres entre deux points choisis à l'aide de la souris (fonction **'distance'**) dans **AnalysisView** (menu **Options**). C'est une fonction très utile pour vérifier l'étalonnage de la caméra et des pixels.

Toutes les mesures en millimètres ne sont bien sûr exactes que si l'étalonnage de **pxlx** et **pxly** est correct (cf **'InitializeBEAM'**).

II.5. 'SaveBEAM' :

A partir du menu **Options**, il est possible de conserver une copie sur disque de l'écran d'affichage et de la matrice **camera**.

Par défaut, les fichiers sont enregistrés dans le répertoire **'c:\temp'**, mais cela peut être modifier dans le sous-programme **'SaveBEAM'**.

La matrice **camera** est sauvegardée sous forme de fichier **TXT** et s'appelle **'M*heure*'** avec pour ***heure***, l'heure à laquelle s'est effectuée la sauvegarde. L'écran est sauvegardé avec le même nom, mais sous format **TIFF**. Un titre est inscrit sur l'écran avec le nom du fichier lors de la sauvegarde de l'écran, car Matlab5 ne sauvegarde pas le titre de la fenêtre.

II.6. 'CloseBEAM' :

'CloseBeam' permet de fermer la fenêtre et de sortir de Matlab tout en libérant la mémoire (les variables sont détruites).

Nous venons de décrire le programme principal **'beamctf'** et son architecture générale. Voyons d'autres points plus spécifiques, utiles pour la programmation.

II.7. La variable PARA :

La variable globale **PARA** est un vecteur de 33 éléments qui contient toutes les références des objets que créé Matlab. La taille de **PARA** est allouée au début de **'beamctf'** dans **'InitializeBEAM'**. Le vecteur est rempli au fur et à mesure que les objets sont créés (dans **'InitializeBEAM'**, **'menubeam'**, **'setframe'**, **'initwind'** et **'initaxes'**).

Voici le contenu de la variable **PARA** :

Numéro	Variable - 'Tag'	Numéro	Variable - 'Tag'
1	fenêtre principale - 'BEAMDIAG'	18	résultat Low - 'lowlabel'
2	menu Play - 'Play2'	19	résultat High - 'highlabel'
3	menu Close - 'Close2'	20	résultat FIT FWHM en X - 'fmarkx'
4	bouton Play - 'Play'	21	résultat FIT FWHM en Y - 'fmarky'
5	bouton Close - 'Close'	22	résultat Pk-Vly en X - 'fmovex'
6	signal de saturation - 'saturationlabel'	23	résultat Pk-Vly en Y - 'fmovey'
7	bouton Ok - 'Okwind'	24	résultat Std Dev en X - 'ffwhhx'
8	axes ViewAxes - 'ViewAxes'	25	résultat Std Dev en Y - 'ffwhhy'
9	axes AnalysisAxes - 'AnalysisAxes'	26	résultat Length en X - 'fwaistx'
10	résultat Mark en X - 'markx'	27	résultat Length en Y - 'fwaisty'
11	résultat Mark en X - 'marky'	28	résultat Mean - 'avglabel'
12	résultat Movement en X - 'movex'	29	résultat RMS - 'rmslabel'
13	résultat Movement en Y - 'movey'	30	image ViewAxes - h-pict dans 'newview'
14	résultat FWHM en X - 'fwhhx'	31	droite souris 'on' - dans 'distance'
15	résultat FWHM en Y - 'fwhhy'	32	droite souris 'off' - dans 'distance'
16	résultat Diameter en X - 'waistx'	33	résultat Barlow - 'textbarl'
17	résultat Diameter en Y - 'waisty'		

II.8. La variable BEAM_DAT :

La variable globale **BEAM_DAT** est un vecteur de 12 éléments qui contient les indicateurs des différentes options de l'analyse. En fonction de la valeur de la variable, tel ou tel calcul est effectué ou tel ou tel graphique est affiché. **BEAM_DAT** est défini dans **'menubeam'** où sont attribuées les valeurs par défaut des différents indicateurs.

Voici la liste des correspondances pour chaque indicateur (en gras, les valeurs par défaut):

Indicateur	Variable	Valeur	Correspondance
1	ViewAxes	0	pas d'affichage
		1	image camera
		2	**image normalized**
2	AnalysisView	0	pas d'affichage
		1	**contour**
		2	3D
3	Croix bleue (cross)	0	pas d'affichage
		1	**barycentre**
		2	maximum
4	Profils (crossp)	0	pas d'affichage
		1	**barycentre**
		2	maximum
		3	point particulier
5	Integrated profiles	0	pas d'affichage
		1	**profils intégrés**
6	Barycentre	0	seuils manuels
		1	**seuils automatiques**
7	Personnal	0	**attente première acquisition**
		1	ok pour acquisition
8	Fit	0	pas de calcul
		1	profils
		2	**profils intégrés**
9	Width	0	profils
		1	**profils intégrés**
10	Mark Reference	0	référence axes
		1	**référence croix**
11	Background	0	pas de calcul
		1	**soustraction**
12	3D Style	0	**style lignes (mesh)**
		1	style facettes (surf)

II.9. Définition de la région d'intérêt :

La région d'intérêt désigne la zone de la matrice ccd de la caméra sélectionnée pour l'image.

Les dimensions et la forme de cette zone dépendent des dimensions affectées aux pixels (cf. **pxlx** et **pxly** dans **'InitializeBEAM'**). La largeur et la hauteur de la région d'intérêt sont calculées en pixels à l'aide des fonctions **'width'** et **'height'** à partir de la valeur de la diagonale (**WINDOW(3)**).

La position et la taille de la région d'intérêt sur la matrice ccd peuvent être modifiées à partir de la fenêtre de contrôle **WINDOW**. Modifier la région d'intérêt interrompt immédiatement les acquisitions et désactive le bouton **Play**. Pour réactiver le bouton **Play**, il faut valider la modification de la région d'intérêt en appuyant sur le bouton **Ok** de la fenêtre **WINDOW**.

En validant la région d'intérêt, on redéfinit la région d'intérêt sur le ccd (fonction **'init'** dans **'setwind'**), et on dessine à nouveau les axes **ViewAxes** (fonction **'newview'**) et **AnalysisAxes** (fonction **'newcont'**).

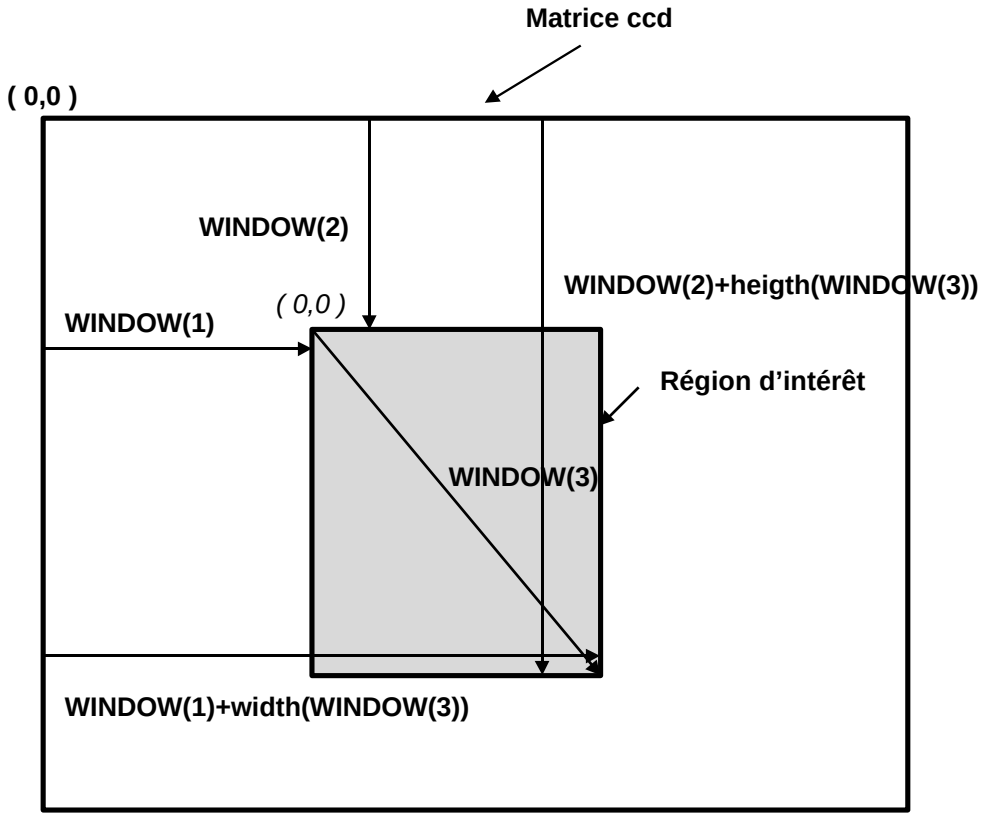

fig 6 : Paramètres de la région d'intérêt.

Ce sont les paramètres qui sont passés à la fonction **'init'** lors de la définition de la région d'intérêt. La variable globale **WINDOW** contient la position et la taille de la région d'intérêt en pixels. Il faut faire attention à l'orientation de chaque repère. Elle est souvent spécifique à un repère.

Les valeurs indiquées dans la fenêtre **WINDOW** sont la position en pixels du centre de la région d'intérêt et la longueur de la diagonale. Le repère de la matrice ccd utilisé ne correspond pas à celui de la fonction **'init'** (ici, origine en bas à gauche et non en haut à gauche),.

II.10. Définition des repères ViewAxes et AnalysisAxes :

Les deux systèmes d'axes sont entièrement définis dans la fonction **'initaxes'** (sous-programme **'InitialiteBEAM'**) et sont redéfinis à chaque affichage d'image pour conserver les proportions de l'image (**'DataAspectRatio'**).

Les repères à l'écran sont toujours orientés de la même manière avec l'origine en bas à gauche, mais cela ne correspond pas à la façon usuelle dont travaille Matlab avec des images (origine en haut à gauche).

Les graduations sont définies dans la fonction **'graduate'** pour avoir une échelle en millimètres. La fonction **'graduate'** peut donner un message d'erreur pour des dimensions de pixels particulières. Il faut alors changer la précision avec laquelle l'échelle est calculée dans **'graduate'**. Ceci ne se produit qu'avec des valeurs peu réalistes (0,12 mm pour un pixel, par exemple).

Les variables globales **ROW** et **COLUMN** contiennent le nombre de lignes et de colonnes de l'image à analyser. La taille de **ViewAxes** correspond exactement aux dimensions de l'image, par contre, **AnalysisAxes** (mode **Contour**) est plus grand (**5*ROW/4** × **5*COLUMN/4**) car les profils sont affichés dans le même repère, mais en dehors de l'image.

Pour le mode **3D**, **AnalysisAxes** a une taille spécifique (cf. les variables **column** et **row**) pour augmenter la vitesse d'affichage en diminuant le nombre de points traités.

III. Utilisation :

Ce paragraphe reprend l'aide disponible à partir du logiciel disponible dans le menu **Help**. Il présente les différentes fonctions disponibles dans le programme.

III.1. Organisation de l'affichage :

L'écran est divisé en plusieurs régions (cf. fig 5).

Le titre de la fenêtre contient le nom de la caméra utilisée et la date de mise en route du programme.

Sous le titre, se trouve la barre de menu. Les lettres soulignées correspondent aux raccourcis de frappe.

La fenêtre **VIDEO LEVELS** indique les différents niveaux de l'image provenant de la caméra (matrice **camera**, pas de normalisation, pas de filtrage). **Low** donne la valeur du pixel de plus petit niveau et **High** la valeur du pixel de plus haut niveau. **Average** et **RMS** donnent la moyenne et la déviation standard des niveaux des pixels de l'image. Les valeurs sont données en pourcentage, 100% correspondant à la réponse la plus élevée (pixel saturé). Un signal de **SATURATION** apparaît si le niveau des pixels est trop élevé pour faire une analyse correcte ou dangereux pour la durée de vie de la caméra. Le signal de saturation peut ne pas apparaître bien que le ccd soit saturé. Cela dépend du seuil de saturation (dans le programme) et de la longueur d'onde d'analyse.

Dans la fenêtre **POSITION**, sont données les mesures géométriques du faisceau, en millimètres, si l'étalonnage est correct. **Mark** indique la position de la croix bleue si elle est affichée dans **AnalysisView**, ou du barycentre sinon. Le point de référence est soit l'origine de l'axe, soit la croix noire (**Mark Reference**) qui est affichée dans **AnalysisView**. Le choix se fait à partir de **Mark Reference** dans le menu **Options**. Il est possible d'initialiser la position de la croix noire en appuyant sur le bouton **Init**. Le nouveau point de référence est alors la position de la croix bleue si

elle est affichée, sinon la position du barycentre ou le centre de la croix rose si c'est le mode **Personal** qui est choisi pour les profils.

Movement indique le déplacement de la croix bleue si elle est affichée, sinon du barycentre. **FWHM** (Full Width at Half Maximum) et **Diameter** donnent la largeur des profils respectivement à 50% et 14% ($1/e^2$) du maximum de l'image ou du profil si c'est le profil intégré. La mesure peut être faite soit sur le profil (bleu), soit sur le profil intégré (orange) (choix dans **Options**). Seuls les profils intégrés sont normalisés. Le maximum est alors le maximum du profil intégré, sinon il s'agit du maximum de l'image (**normalized**). Le calcul de la largeur est expliqué dans le paragraphe **II.4** (**Largeur**) de cette note. Le choix entre le profil et le profil intégré pour le calcul de la largeur dépend de la forme du faisceau à analyser. Il faut choisir le mode de calcul le mieux adapté à la situation.

La fenêtre de **FIT** donne les résultats du fit des profils. Le modèle de fit est pour le moment gaussien, mais il est possible d'utiliser un autre modèle (à définir). **FWHM** donne la largeur à mi-hauteur en millimètres de la gaussienne la plus proche (au sens des moindres carrés) du profil. Le choix du profil se fait dans le menu **Options**. L'écart maximal (max - min) entre le fit et le profil est donné par **Pk-Vly** (Peak to Valley) et la déviation standard par **Std Dev**. Ces valeurs sont multipliées par 100 pour l'affichage.

La fenêtre **Length** donne la distance en millimètres entre deux points choisis dans **AnalysisView** à l'aide de la souris. Le résultat est décomposé suivant X et Y.

III.2. Les commandes :

La fenêtre **WINDOW** permet la sélection de la région d'intérêt dans la matrice ccd. La longueur de la diagonale en pixels est donné par **Part**. La position du centre de la région d'intérêt et donnée dans **Up**/**Down** et **Left**/**Right**. Le bouton **Ok** permet de valider le choix de la région d'intérêt et de reprendre les acquisitions.

CntTh permet de modifier le seuil du dernier contour (rouge) pour l'analyse du faisceau. Le choix du seuil se fait entre 55% et 99%.

BarLow et **BarHigh** permettent de modifier les seuils bas et haut pour le calcul du barycentre. **BarLow** n'est pris en compte que si le mode de calcul des seuils n'est pas automatique (choix dans **Options**). Le seuil bas est indiqué dans la fenêtre **BarLow**.

Les boutons **Play**, **Stop** et **Close** permettent de lancer et d'arrêter les acquisitions, ainsi que de quitter le programme.

III.3. Le menu View :

Il se rapporte à **ViewAxes**. Il permet de choisir l'affichage de l'image provenant de la caméra (matrice **camera**) avec **Video**, de la matrice normalisée (**normalized**) avec Normalized ou d'aucune image avec **Off**.

III.4. Le menu Analysis :

Il se rapporte à la partie centrale d'**AnalysisView**. Il permet de choisir entre l'affichage des contours avec **Contour**, ou de l'image en 3 dimensions avec **3D** ou d'aucune analyse avec **Off**.

Les contours présentent trois courbes de niveau : en bleu à 14% du maximum de l'image, en vert à 50% et en rouge au niveau indiqué par **CntTh** (80% par défaut). Si l'image est très bruitée, il est possible avec la fonction **Background** (menu **Options**) de soustraire le bruit de fond (estimé) à l'image. Dans ce cas, l'affichage des contours est plus rapide.

La seconde partie du menu concerne le repérage d'un point caractéristique du faisceau. Ce point peut être le barycentre de l'image avec **Barycentre** ou le maximum avec **Maximum**, ou aucun point avec **Off**. Le point est repéré par une croix bleue.

III.5. Le menu Profile :

Il se rapporte aux parties extérieures d'**AnalysisView** où sont affichés les différents profils.

Le code de couleurs pour les profils est le suivant : courbe bleue pour le profil (section), courbe orange pour le profil intégré, courbe rose pour le profil du fit.

La première partie du menu concerne le profil (bleu). Le point de référence, à partir duquel sont réalisés les profils en X et Y, peut être le barycentre avec **Barycentre**, le maximum de l'image avec **Maximum**, ou un point particulier choisi avec la souris dans **AnalysisView** avec **Personal**. Aucun profil n'est affiché si **Off** est sélectionné.

La deuxième partie du menu permet de choisir d'afficher ou non le profil intégré avec **Integrated**. Le profil intégré est normalisé et son décalage (offset) est compensé pour l'affichage ainsi que pour les calculs de largeur.

D'autre part, s'il y a beaucoup de lumière parasite, il peut être nécessaire d'utiliser la fonction **Background** (menu **Options**) pour faire des mesures correctes de largeur sur les profils.

III.6. Le menu Options :

Le choix du mode de calcul pour les seuils du barycentre se fait dans **Barycentre**. **Manual** permet de choisir à l'aide de **BarLow** et **BarHigh** les seuils pour le calcul. Avec **Automatic**, **BarLow** est calculé automatiquement en fonction des niveaux de l'image.

Dans **Widths**, on choisit à partir de quel profil le calcul de largeur est effectué, profil avec **Profiles** et profil intégré avec **Integrated**.

Background permet de soustraire le bruit de fond de l'image à la matrice **normalized**.

Mark Reference permet de changer le point de référence à partir duquel est indiquée la position de la croix bleue ou du barycentre. Quand ce mode est sélectionné, le point de référence devient la croix noire, sinon c'est l'origine du repère.

Length est une fonction qui permet de mesurer directement à l'écran à l'aide de la souris une distance en millimètres. Elle facilite, entre autres, l'opération d'étalonnage de chaque caméra.

Fit permet de choisir le profil que l'on désire comparer au modèle gaussien (seul modèle disponible pour le moment). La fonction **Fit** est inactive si **Off** est sélectionné, sinon **Profiles** permet de travailler avec les profils (bleus) et **Integrated** avec les profils intégrés. Les résultats apparaissent dans la fenêtre **FIT** et les profils dans **AnalysisView**, superposés aux autres profils.

Print lance l'impression de l'écran sur l'imprimante sélectionnée. L'impression se fait automatiquement sur un format A4 en mode paysage. Tous les paramètres sont définis par Matlab. Pour imprimer différemment, il faut modifier les propriétés de la fenêtre **'BEAMDIAG'**.

Save permet la sauvegarde de l'écran dans le fichier **'c:\temp\M*heure*.tif'** et de la matrice image (**camera**) dans le fichier **'c:\temp\M*heure*.txt'**. ***heure*** est l'heure à laquelle est effectuée la sauvegarde.

III.7. Le menu Help :

Il donne une page d'aide sur chaque partie du programme.

IV. Conclusion :

Nous venons de décrire brièvement l'architecture du programme **BEAMDIAG** et les principales fonctions disponibles.

Il est possible de modifier le programme pour intégrer de nouvelles fonctions ou apporter des améliorations. C'est un des grands avantages de ce programme. Il est facilement adaptable.

En annexe, se trouve le listing de toutes les fonctions citées dans la note. Il est ainsi aisé de suivre les commentaires et les références de la note.

L'utilisation du logiciel dans les différents laboratoires où il est installé, donne entière satisfaction. Les analyses du faisceau sont fiables dès que l'étalonnage des caméras est correct. Toutefois la vitesse de rafraîchissement des acquisitions peut être améliorées. Pour cela, il peut être intéressant de réécrire certaines parties du programme en C ou équivalent.

Annexe

Seuls les fichiers cités dans la note se trouvent en annexe. Ils apparaissent par ordre alphabétique.

Backgrnd.m

```
function ans=backgrnd(vect)

% Calculate the background of the vector

high=max(vect);
bg=mean(vect);
level=bg+(high-bg)*0.15;
mask=sparse(vect<level);
nvect=vect.*mask;
bg=sum(nvect)/sum(mask);
level=bg+(high-bg)*0.15;
mask=sparse(vect<level);
nvect=vect.*mask;
bg=sum(nvect)/sum(mask);
ans=vect-bg;
```

Beamctf.m

```
function beamctf(action,numcam,pxlx,pxly)

% BEAMDIAG Beam diagnostics and analysis.
%        DBS. 12_1997


% Callback Routines:
%   InitializeBEAM
%   DisplayBEAM
%   SaveBEAM
%   CloseBEAM

%   PROG 1->CAMERA , 2->ONE SHOT CAMERA , 3->VIEW, 4->COMPUTE

global BEAM_DAT ROW COLUMN PREVIOUS BARTH CONTOURTH ABSCISSA ORDINATE...
             EPSILON WINDOW NUMCAM PXLX PXLY camera PERSONAL DATA PROG     ...
             REF DIAGMAX GRADX GRADY GRADY PARA LOOPLVL

%====================================
% Parameters
%====================================

PROG=4;
LOOPLVL=0.08;

if nargin>1,
 NUMCAM=numcam;
     PXLX=pxlx;
     PXLY=2*pxly;
end;

feval(action);
return;


%=================================================
%  Sub-function - InitializeBEAM -Initialize figure

%=================================================

function InitializeBEAM

global NUMCAM ROW COLUMN PXLX PXLY WINDOW DIAGMAX DATA PROG    ...
      EPSILON BARTH CONTOURTH PERSONAL PREVIOUS REF PARA

% If beamctf is already running, bring it to the foreground.

handle=findobj(allchild(0),'Tag','BEAMDIAG');
if (~isempty(handle)),
  figure(handle(1));
  return
end;

% Name of cameras

laboCTF={'';                                    ...
        'IR Laser Imaging - ';                  ...
        'Drive Beam Virtual Cathode - ';        ...
        'Probe Beam Virtual Cathode - ';        ...
        'Drive Beam UV Camera - '};

Photocathode={'';                               ...
        'Electron Beam Camera - ';              ...
        'Laser Beam Virtual Cathode - ';        ...
        'Microscope - ';                        ...
        'Visualisation - '};

NAMECAMERA=Photocathode;

length=size(NAMECAMERA);
length=length(1);


% Name of pictures

DATA={'camir2';'catuv4';'catuv5';'catuv6';'lablas';'carre1';'Mirep3'};

% Number of colors

screenD=get(0,'ScreenDepth');
if screenD>8,
  colorres=256;
else
  colorres=128;
end;

% Definition of the main figure

h_figure = figure('Units','normalized',         ...
     'Color',[0.87 0.87 0.87],                  ...
     'Colormap',jet(colorres),                  ...
 'DefaultLineLineWidth',1.5,                    ...
 'MenuBar','none',                              ...
     'Name',[char(NAMECAMERA(mod(NUMCAM,length)+1)) date ],
...
```

```
  'NumberTitle','off',                                    ...
  'HandleVisibility','on',                                ...
  'Resize','off',                                        ...
  'BusyAction','Queue',                                     ...
  'Interruptible','off',                                 ...
  'IntegerHandle','off',                                  ...
  'DefaultAxesColorOrder',[0 0.2 1;0.4 1 0.6;1 0 0],            ...
  'Renderer','painters',                                   ...
  'BackingStore','off',                                   ...
  'Position',[0.004 0.04 0.993 0.906],                          ...
  'Tag','BEAMDIAG',                                        ...
  'PaperUnits','centimeters',                              ...
  'PaperType','a4letter',                                  ...
  'PaperOrientation','landscape',                               ...
  'PaperPositionMode','manual',                              ...
  'PaperPosition',[0.5 2.5  25 17],                          ...
  'Visible','off');

  PARA=zeros(1,33);
  PARA(1)=findobj('TAG','BEAMDIAG');


  %====================================
  % Initialize menus
  %====================================


  menubeam;

  %====================================
  % Parameters for the WINDOW
  %====================================


DIAGMAX=min(round(511/2*sqrt(PXLY/PXLX)),round(511*sqrt(PXLX/
PXLY)));
  WINDOW=[round((511-width(DIAGMAX)-1)/2),round((511-
height(DIAGMAX)-1)/2),DIAGMAX];

  if (PROG==1) | (PROG==2),
    warning off;

windinit=init(NUMCAM,WINDOW(1),WINDOW(2),WINDOW(1)+widt
h(WINDOW(3)),       ...
                  WINDOW(2)+height(WINDOW(3)));
  end;

  %====================================
  % Parameters for the AXES
  %====================================

  if (PROG==1) | (PROG==2),
       COLUMN=floor(width(WINDOW(3))+1);
       ROW=floor((height(WINDOW(3))+1)/2);
  elseif PROG==3;
       [X,map]=imread(['j:\data\cern\image\',char(DATA(1)),'.tif'],1);
       [ROW,COLUMN]=size(X);
  else
       COLUMN=floor(width(WINDOW(3))+1);
       ROW=floor((height(WINDOW(3))+1)/2);
  end;

  camera=zeros(ROW,COLUMN);
  PREVIOUS=[COLUMN/2 ROW/2];
  REF=PREVIOUS;
  PERSONAL=PREVIOUS;
  graduate;

  %====================================
  % Parameters
  %====================================

  BARTH=[0.55 0.95];
  EPSILON=0.055;
  CONTOURTH=[0.1353 0.5 0.8];

  %====================================
  % Initialize FRAMES
  %====================================

  setframe;

  %====================================
  % Initialize WINDOW controls
  %====================================

  initwind;

  %====================================
  % Initialize AXES
  %====================================

  initaxes;

  %====================================
  % Uncover the figure
  %====================================

  set(h_figure,'Visible','on');

  return

  %================================================
  %  Sub-function - DisplayBEAM - display pictures
  %================================================

  function DisplayBEAM

  global BEAM_DAT ROW COLUMN PREVIOUS BARTH
CONTOURTH ABSCISSA ORDINATE...
              EPSILON WINDOW NUMCAM PXLX PXLY camera
PERSONAL DATA PROG     ...
              REF DIAGMAX GRADX GRADY GRADY PARA
LOOPLVL

    mapfig=get(PARA(1),'Colormap');
    set(PARA(4),'Userdata',1);
    set(PARA(4),'Enable','off');
    set(PARA(5),'Enable','off');
    set(PARA(2),'Enable','off');
    set(PARA(3),'Enable','off');

    BEAM_DAT(7)=1;
    ViewAxes=PARA(8);
    AnalysisAxes=PARA(9);
    num=0;
    length=size(DATA);
    length=length(1);

    row=min(ROW,35);
    column=min(COLUMN,35);
        gradX=GRADX*column/COLUMN;
    gradY=GRADY*row/ROW;

    %============================================
    % Main loop
    %============================================

    while  get(PARA(4),'Userdata')==1,


       % Load original image
       %===================

  if (PROG==1) | (PROG==2),

     acqvideo;
        !c:\w95\profiles\local\desktop\oddtra~1.pif
```

```
    [X,map]=imread('c:\video\images\image.tif',1);
    camera=ind2gray(X,gray(256));

      elseif PROG==3
            num=num+1;
            [X,map]=imread(['j:\data\cern\image\'
char(DATA(mod(num,length)+1)) '.tif'],1);
    [ROW,COLUMN]=size(X);
    camera=ind2gray(X,gray(256));
            graduate;

      else
    camera=simbeam(NUMCAM);
      end;


      % Levels
      %=======

      LowLevel=min(min(camera));
      HighLevel=max(max(camera));
      AvgLevel=mean(mean(camera));
      RMS=std2(camera);


 % Test on levels
 %===============

 if (HighLevel-LowLevel)>LOOPLVL,

      % Normalisation
      %==============

      if LowLevel < HighLevel
            normalised=(camera-LowLevel)/(HighLevel-LowLevel);
      else
            normalised=camera-LowLevel;
      end

      % Draw ViewAxes
      %==============

      if BEAM_DAT(1)==1,
            picture=camera;
      else
            picture=normalised;
      end;
      if BEAM_DAT(1)>0
        axes(ViewAxes);
        set(PARA(30),              ...
          'CData',picture,         ...
          'Erasemode','none');
      end;

      % Analysis View
      %==============

      % Background
      %===========

    if BEAM_DAT(11)>0,
    normalised=noise(normalised);
    end;

      % Barycentre
      %===========

      if (BEAM_DAT(3)<2) | (BEAM_DAT(4)==1),
            if BEAM_DAT(6)==1,
                  avg=mean(mean(normalised));
                  RMS2=std2(normalised);
                  barth=min(avg+5*RMS2*(BARTH(2)-avg)^3,0.8);
            else
                  barth=BARTH(1);
            end;
            mask=sparse(normalised>barth)-
sparse(normalised>BARTH(2));
            matr=normalised.*mask;
            [matX,matY]=meshgrid(1:COLUMN,1:ROW);
      tot=sum(sum(matr));
      if tot~=0,
        barycentre=round(full([sum(sum(matr.*matX))/tot
sum(sum(matr.*matY))/tot]));
      else
        barycentre=0;
      end;
            if ~isempty(find(barycentre==0))
                  barycentre=[];
      end;
      set(PARA(33),'String',sprintf('%3.0f',round(100*barth)));
        end;

        % Maximum
        %========

      if (BEAM_DAT(3)==2) | (BEAM_DAT(4)==2),
            [maxX,maxY]=find(normalised>0.92);
            maximum=[median(maxY) median(maxX)];
  end;


  % AnalysisAxes
  %==============

  axes(AnalysisAxes);
  set(AnalysisAxes,'NextPlot','replace');

      % 3D View
      %========

  if BEAM_DAT(2)==2,

    resized=imresize(normalised,[row,column],'nearest');
    if BEAM_DAT(12)>0
      h_pict=surf(resized);
    else
      h_pict=mesh(resized);
    end;
      set(AnalysisAxes,                                  ...
       'Units', 'normalized',                            ...
       'Position', [0.485 0.26 0.49 0.68],               ...
       'TickDir','out',                                  ...
       'XLim',[1 column],                                ...
              'XTick', [1 gradX],                        ...
       'XTickLabel',ABSCISSA,                            ...
       'FontSize',11,                                    ...
       'FontWeight','bold',                              ...
       'XGrid','on',                                     ...
       'YDir','reverse',                                 ...
       'YLim',[1 row],                                   ...
       'YTick',gradY,                                    ...
       'YTickLabel',ORDINATE,                            ...
       'YGrid','on',                                     ...
       'DataAspectRatioMode','manual',                   ...
       'DataAspectRatio',[1 row/column 0.05],            ...
       'ZLim',[0 1],                                     ...
       'ZGrid','on',                                     ...
       'ZTickLabel','',                                  ...
       'Tag','AnalysisAxes',                             ...
       'NextPlot','add');
            View([-40 30]);

    if BEAM_DAT(3)==2,
            cross=maximum;
      else
            cross=barycentre;
      end;
```

```
            crossp=cross;
            if BEAM_DAT(3)>0 & ~isempty(cross)
                crossd(2)=cross(2)*row/ROW;
                crossd(1)=cross(1)*column/COLUMN;
                plot3([1 column],[crossd(2) crossd(2)],[1 1],'Color',[0 0
0.8]);
                plot3([crossd(1) crossd(1)],[1 row],[1 1],'Color',[0 0
0.8]);
                if BEAM_DAT(10)==1
                    plot3([max(REF(1)-
COLUMN/10,1)*column/COLUMN
min(REF(1)+COLUMN/10,COLUMN)*column/COLUMN],...
                        [REF(2)*row/ROW REF(2)*row/ROW],[1
1],'k');
                    plot3([REF(1)*column/COLUMN
REF(1)*column/COLUMN],...
                        [max(REF(2)-ROW/10,1)*row/ROW
min(REF(2)+ROW/10,ROW)*row/ROW],...
                        [1 1],'k');
                end;
            end;
        if (BEAM_DAT(9)==1) | (BEAM_DAT(8)==2),
            Xintegrated=backgrnd(sum(normalised'));
            Yintegrated=backgrnd(sum(normalised));
        end;
    else

        % Contour View
        %=============


            if BEAM_DAT(2)==1,
         [C,h_pict]=contour(normalised,CONTOURTH,'-');
         set(h_pict,'Erasemode','none');
        set(AnalysisAxes,'NextPlot','add');
    end;

    plot([1 5*COLUMN/4],[0 0],'black');

    set(AnalysisAxes,                                   ...
            'Units', 'normalized',                          ...
    'Position', [0.37 0.24 0.74 0.74],                  ...
    'TickDir','in',                                 ...
    'XLim',[1 5*COLUMN/4],                              ...
    'XTick', [1 GRADX],                             ...
    'XTickLabel',ABSCISSA,                              ...
    'FontSize',11,                                  ...
    'FontWeight','bold',                            ...
    'XGrid','on',                                   ...
    'YDir','reverse',                               ...
    'YLim',[-ROW/4 ROW],                                ...
    'YTick',GRADY,                                  ...
    'YTickLabel',ORDINATE,                              ...
    'YGrid','on',                                   ...
    'DataAspectRatioMode','manual',                     ...
    'DataAspectRatio',[1 ROW/COLUMN 1],                 ...
            'Tag','AnalysisAxes',                       ...
    'NextPlot','add');

        plot([COLUMN COLUMN],[-ROW/4 ROW],'black');


        % Cross
        %======


            if BEAM_DAT(3)==2,
                cross=maximum;
            else
                cross=barycentre;
            end;
            if (BEAM_DAT(3)>0 & ~isempty(cross))
                plot([1 COLUMN],[cross(2) cross(2)],'Color',[0 0 0.8]);
                plot([cross(1) cross(1)],[0 ROW],'Color',[0 0 0.8]);
                if BEAM_DAT(10)==1
                    plot([max(REF(1)-COLUMN/10,1)
min(REF(1)+COLUMN/10,COLUMN)],[REF(2) REF(2)],...
                    'k');
                    plot([REF(1) REF(1)],[max(REF(2)-ROW/10,1)
min(REF(2)+ROW/10,ROW)],...
                    'k');
                end;
            end;



        % Profile
        %========

            if BEAM_DAT(4)==1,
                crossp=barycentre;
            elseif BEAM_DAT(4)==2
                crossp=maximum;
            elseif BEAM_DAT(4)==3
                crossp=PERSONAL;
                plot([max(crossp(1)-COLUMN/10,1)
min(crossp(1)+COLUMN/10,COLUMN)],[crossp(2) crossp(2)],...
                    'm');
                plot([crossp(1) crossp(1)],[max(crossp(2)-ROW/10,1)
min(crossp(2)+ROW/10,ROW)],...
                    'm');

            else
                crossp=cross;
            end;
            if (BEAM_DAT(4)>0 & ~isempty(crossp)),
                Xprofile=(normalised(:,crossp(1))/4+1)*COLUMN;
                Yprofile=-normalised(crossp(2),:)*ROW/4;
                plot(Xprofile,1:ROW,'Color',[0 0 0.8]);
                plot(Yprofile,'Color',[0 0 0.8]);
            end;

        % Integrated
        %===========

            if (BEAM_DAT(5)==1) | (BEAM_DAT(9)==1) |
(BEAM_DAT(8)==2),
                Xintegrated=backgrnd(sum(normalised'));
                Yintegrated=backgrnd(sum(normalised));
         if BEAM_DAT(5)==1,
           if max(Xintegrated)~=0,
             Xprofile=((Xintegrated/max(Xintegrated))/4+1)*COLUMN;
             plot(Xprofile,1:ROW,'Color',[1 0.5 0]);
           end;
           if max(Yintegrated)~=0,
                    Yprofile=-
(Yintegrated/max(Yintegrated))*ROW/4;
             plot(Yprofile,'Color',[1 0.5 0]);
           end;
         end;
            end;
        end;

        % Width
        %======

        if (~isempty(crossp)) | (BEAM_DAT(9)==1),
            if (BEAM_DAT(9)==1),
                maxXw=max(Yintegrated);
                if maxXw>0
           Xw=Yintegrated/maxXw;
         else
           Xw=zeros(1,COLUMN);
                end;
                maxYw=max(Xintegrated);
         if maxYw>0
           Yw=Xintegrated/maxYw;
```

```
        else
          Yw=zeros(1,ROW);
                end;
            else
                Xw=normalised(crossp(2),:);
                Yw=(normalised(:,crossp(1)))';
            end;
            Xwidth1=contourc([Xw;Xw],[CONTOURTH(2)
CONTOURTH(2)]);
            if ~isempty(Xwidth1)
                Xcol=Xwidth1(2,:)==1;
                Xwidth=Xwidth1(1,Xcol);
            else
                Xwidth=[];
            end;
            Ywidth1=contourc([Yw;Yw],[CONTOURTH(2)
CONTOURTH(2)]);
            if ~isempty(Ywidth1)
                Ycol=Ywidth1(2,:)==1;
                Ywidth=Ywidth1(1,Ycol);
            else
                Ywidth=[];
            end;
            MaxX=max(Xw);
            MaxY=max(Yw);
    MaxX=find(Xw==MaxX);
            MaxY=find(Yw==MaxY);

            Xleft=MaxX(1)-Xwidth;
            Xleft=min(Xleft(Xleft>0));
            Xright=Xwidth-MaxX(1);
            Xright=min(Xright(Xright>0));
            Yup=MaxY(1)-Ywidth;
            Yup=min(Yup(Yup>0));
            Ydown=Ywidth-MaxY(1);
            Ydown=min(Ydown(Ydown>0));

      else
            Xleft=[];
            Xright=[];
            Yup=[];
            Ydown=[];
      end;
      if isempty(Xleft) | isempty(Xright)
            Xwidth='';
      else
            Xwidth=(round(10*scaleX(Xright+Xleft)))/10;
      end;
      if isempty(Yup) | isempty(Ydown)
            Ywidth='';
      else
            Ywidth=(round(10*scaleY(Ydown+Yup)))/10;
      end;

      % Waist
      %======

      if (~isempty(crossp)) | (BEAM_DAT(9)==1),
            if (BEAM_DAT(9)==1),
                maxXw=max(Yintegrated);
                if maxXw>0
        Xw=Yintegrated/maxXw;
       else
          Xw=zeros(1,COLUMN);
                end;
                maxYw=max(Xintegrated);
       if maxYw>0
          Yw=Xintegrated/maxYw;
       else
          Yw=zeros(1,ROW);
       end;
    else
                Xw=normalised(crossp(2),:);
                Yw=(normalised(:,crossp(1)))';
                end;
                Xwaist1=contourc([Xw;Xw],[CONTOURTH(1)
CONTOURTH(1)]);
                if ~isempty(Xwaist1)
                    Xcol=Xwaist1(2,:)==1;
                    Xwaist=Xwaist1(1,Xcol);
                else
                    Xwaist=[];
                end;
                Ywaist1=contourc([Yw;Yw],[CONTOURTH(1)
CONTOURTH(1)]);
                if ~isempty(Ywaist1)
                    Ycol=Ywaist1(2,:)==1;
                    Ywaist=Ywaist1(1,Ycol);
                else
                    Ywaist=[];
                end;
                MaxX=max(Xw);
                MaxY=max(Yw);
      MaxX=find(Xw==MaxX);
                MaxY=find(Yw==MaxY);
                Xleft=MaxX(1)-Xwaist;
                Xleft=min(Xleft(Xleft>0));
                Xright=Xwaist-MaxX(1);
                Xright=min(Xright(Xright>0));
                Yup=MaxY(1)-Ywaist;
                Yup=min(Yup(Yup>0));
                Ydown=Ywaist-MaxY(1);
                Ydown=min(Ydown(Ydown>0));

        else
                Xleft=[];
                Xright=[];
                Yup=[];
                Ydown=[];
        end;
        if isempty(Xleft) | isempty(Xright)
                Xwaist='';
        else
                Xwaist=(round(10*scaleX(Xright+Xleft)))/10;
        end;
        if isempty(Yup) | isempty(Ydown)
                Ywaist='';
        else
                Ywaist=(round(10*scaleY(Ydown+Yup)))/10;
        end;

    % Fit
    %===========

    if ((~isempty(crossp)) & (BEAM_DAT(8)==1)) | (BEAM_DAT(8)==2);
      if (BEAM_DAT(8)==2),
        if ~isempty(Xleft) & ~isempty(Xright),
                    maxXw=max(Yintegrated);
                    if maxXw>0
            Xw=Yintegrated/maxXw;
           else
            Xw=zeros(1,COLUMN);
          end;
        end;
        if ~isempty(Ydown) & ~isempty(Yup),
                      maxYw=max(Xintegrated);
          if maxYw>0
            Yw=Xintegrated/maxYw;
          else
            Yw=zeros(1,ROW);
          end;
         end;
      else
                    Xw=normalised(crossp(2),:);
                    Yw=(normalised(:,crossp(1)))';
                end;

      if ~isempty(Xleft) & ~isempty(Xright),
```

```
    [Xbeta,Xpv,Xdev]=fitting(Xw,Xleft+Xright);
     Xbeta=reshape(Xbeta,[1 4]);
  else
    Xbeta=[];
    Xpv='';
    Xdev='';
  end;
  if ~isempty(Ydown) & ~isempty(Yup),
    [Ybeta,Ypv,Ydev]=fitting(Yw,Ydown+Yup);
    Ybeta=reshape(Ybeta,[1 4]);
  else
    Ybeta=[];
    Ypv='';
    Ydev='';
  end;

  % Graphs


  if (BEAM_DAT(4)>0) | (BEAM_DAT(5)>0) ,
    if ~isempty(Ydown) & ~isempty(Yup) & ~isempty(Ybeta),
       Xprofile=((gausbeam(Ybeta,ROW))/4+1)*COLUMN;
       plot(Xprofile,1:ROW,'Color','m');
    end;
    if ~isempty(Xleft) & ~isempty(Xright) & ~isempty(Xbeta),
       Yprofile=-((gausbeam(Xbeta,COLUMN)))*ROW/4;
       plot(Yprofile,'Color','m');
    end;
  end;

  % Results

  if isempty(Xbeta),
    Xww='';
  else
    Xww=Xbeta(4);
  end;
  ffwhhx=(round(10*scaleX(sqrt(log(2)/2)*Xww)))/10;
  if isempty(Ybeta),
    Yww='';
  else
    Yww=Ybeta(4);
  end;
  ffwhhy=(round(10*scaleY(sqrt(log(2)/2)*Yww)))/10;
  fwidthx=1e2*Xpv;
  fwidthy=1e2*Ypv;
  ferrx=1e2*Xdev;
  ferry=1e2*Ydev;

 else

   ffwhhx='';
   ffwhhy='';
   fwidthx='';
   fwidthy='';
   ferrx='';
   ferry='';

 end;

    % Saturation
    %===========

SaturationTh=0.95;
SaturedPixels=20;

    Saturation=sparse(camera>SaturationTh);
    if sum(sum(Saturation))>SaturedPixels
          set(PARA(6),'Visible','on');
    else
          set(PARA(6),'Visible','off')
    end;

    % Movements
    %==================

    if ~isempty(cross)
          Movex=(round(10*scaleX(cross(1)-PREVIOUS(1))))/10;
          Movey=(round(10*scaleY(PREVIOUS(2)-cross(2))))/10;
          PREVIOUS=cross;
          if BEAM_DAT(10)==1
            Markx=(round(10*scaleX(cross(1)-REF(1))))/10;
            Marky=(round(10*scaleY(REF(2)-cross(2))))/10;
          else
            Markx=(round(10*scaleX(cross(1))))/10;
            Marky=(round(10*scaleY(ROW-cross(2))))/10;
          end;
    else
          Movex='';
          Movey='';
          PREVIOUS=[COLUMN/2 ROW/2];
          Markx='';
          Marky='';
  end;

  % Uncover pictures
    %==================

  drawnow;

  % Level results
    %==============

    set(PARA(18),'String',sprintf('%3.0f',round(100*LowLevel)));
    set(PARA(19),'String',sprintf('%3.0f',round(100*HighLevel)));
    set(PARA(28),'String',sprintf('%3.0f',round(100*AvgLevel)));
    set(PARA(29),'String',sprintf('%3.0f',round(100*RMS)));

  % Position results
    %==================

  set(PARA(10),'String',sprintf('%2.1f',Markx));
    set(PARA(11),'String',sprintf('%2.1f',Marky));
    set(PARA(12),'String',sprintf('%2.1f',Movex));
    set(PARA(13),'String',sprintf('%2.1f',Movey));
    set(PARA(14),'String',sprintf('%2.1f',Xwidth));
    set(PARA(15),'String',sprintf('%2.1f',Ywidth));
    set(PARA(16),'String',sprintf('%2.1f',Xwaist));
    set(PARA(17),'String',sprintf('%2.1f',Ywaist));

  % Fit results
  %==================

    set(PARA(20),'String',sprintf('%2.1f',ffwhhx));
    set(PARA(21),'String',sprintf('%2.1f',ffwhhy));
    set(PARA(22),'String',sprintf('%3.0f',fwidthx));
    set(PARA(23),'String',sprintf('%3.0f',fwidthy));
    set(PARA(24),'String',sprintf('%3.0f',ferrx));
    set(PARA(25),'String',sprintf('%3.0f',ferry));

else

  % Level results
  %==================

    set(PARA(18),'String',sprintf('%3.0f',round(100*LowLevel)));
    set(PARA(19),'String',sprintf('%3.0f',round(100*HighLevel)));
    set(PARA(28),'String',sprintf('%3.0f',round(100*AvgLevel)));
    set(PARA(29),'String',sprintf('%3.0f',round(100*RMS)));
    pause(0.2);
  end;

  if PROG==2,
    set(PARA(4),'Userdata',0)
  end;
```

```
    end;

    if strcmp(get(PARA(7),'Enable'),'off'),
        set(PARA(4),'Enable','on');
        set(PARA(2),'Enable','on');
    end;
    set(PARA(5),'Enable','on');
    set(PARA(3),'Enable','on');

  return



%================================================
  %  Sub-function - SaveBEAM - Save data

%================================================

  function SaveBEAM

  global camera PARA

  clck=clock;
  clckstr=[int2str(clck(4)),int2str(clck(5)),int2str(clck(6))];
  str=['c:\temp\M',clckstr,'.txt'];
  str2=['c:\temp\M',clckstr,'.tif'];
  save(str,'camera','-ascii','-tabs');
  h_text=uicontrol(                                   ...
       'Parent',PARA(1),                                ...
   'Style','text',                                  ...
       'Units','normalized',                            ...
       'Position',[0.015 0.214 0.45 0.023],                 ...
   'String',[get(PARA(1),'Name'),' - ',datestr(now,13),' - M',clckstr],...
   'FontSize',10,                                   ...
   'FontWeight','bold',                             ...
   'ForegroundColor','b',                               ...
   'BackgroundColor',[0.87 0.87 0.87],                  ...
   'Visible','on');

   [X,map]=capture(PARA(1));
   imwrite(X,map,str2);

   set(h_text,'Visible','off');


  return

  %================================================
  %  Sub-function - CloseBEAM - Close figure
  %================================================

  function CloseBEAM

  global BEAM_DAT ROW COLUMN PREVIOUS BARTH
CONTOURTH ABSCISSA ORDINATE   ...
              EPSILON WINDOW NUMCAM PXLX PXLY camera
PERSONAL DATA PROG       ...
              REF DIAGMAX GRADX GRADY GRADY PARA
LOOPLVL

    close(PARA(1));
    clear global BEAM_DAT ROW COLUMN PREVIOUS BARTH
CONTOURTH ABSCISSA ORDINATE PARA LOOPLVL...
         EPSILON WINDOW NUMCAM PXLX PXLY camera
PERSONAL DATA PROG REF DIAGMAX GRADX GRADY;
    warning on;



  return
```

Dblaseruv.m

```
    beamctf('InitializeBEAM',1,0.013,0.013);
```

Dblaseruv.wi2

```
    copy "c:\\matlab\\beam\\dblasuv.m" "c:\\matlab\\startup.m"
    execute /MINIMIZED "c:\\matlab\\bin\\matlab.exe"
```

Distance.m

```
    function distance(varargin)

% Mesure the distance (X,Y) between two points

global ROW COLUMN PARA LINE_X LINE_Y



if ((length(varargin) >= 1) & isstr(varargin{1}))
     feval(varargin{:});
 return;
end

LINE_X = [0 0];
LINE_Y = [0 0];

set(PARA(1),'Pointer','crosshair');
set(PARA(1),'WindowButtonDownFcn','distance(''FirstButtonDown'');');
axes(PARA(9));

    PARA(31) = line('XData', LINE_X, ...
                'YData', LINE_Y, ...
                'UserData','', ...
                'Visible', 'off', ...
                'Clipping', 'off', ...
                'Color', 'k', ...
                'LineStyle', '-', ...
                'EraseMode', 'xor');

    PARA(32) = line('XData', LINE_X, ...
                'YData', LINE_Y, ...
                'Visible', 'off', ...
                'Clipping', 'off', ...
                'Color', 'w', ...
                'LineStyle', '--', ...
                'EraseMode', 'xor');

    eval('waitfor(PARA(31), ''UserData'', ''Completed'')');
```

```
   x = LINE_X(:);
   y = LINE_Y(:);

     if (isempty(x))
       x = zeros(1,2);
     end
     if (isempty(y))
       y = zeros(1,2);
     end


% Restore the figure's initial state
if (ishandle(PARA(1)))
    set(PARA(1), 'WindowButtonDownFcn', '', ...
                 'WindowButtonMotionFcn', '', ...
                 'WindowButtonUpFcn', '', ...
                 'Pointer', 'arrow', ...
                 'Interruptible', 'off');
end

% Clean up the global workspace
clear global LINE_X LINE_Y

% Return the answer
    Lx=(round(10*scaleX(x(2)-x(1))))/10;
    Ly=(round(10*scaleY(y(1)-y(2))))/10;
                        set(PARA(26),'String',sprintf('%2.1f',Lx));
                        set(PARA(27),'String',sprintf('%2.1f',Ly));

%-----------------------------------------------------
% Subfunction FirstButtonDown
%-----------------------------------------------------
function FirstButtonDown

global ROW COLUMN PARA LINE_X LINE_Y


pt = get(PARA(9), 'CurrentPoint');

if pt(1,1)>=1 & pt(1,1)<=COLUMN & pt(1,2)>=1 & pt(1,2)<=ROW,
   LINE_X(1)=pt(1,1);
   LINE_Y(1)=pt(1,2);
else return;
end;
set([PARA(31) PARA(32)], ...
        'XData', pt(1,1), ...
        'YData', pt(1,2), ...
        'Visible', 'on');

% Let the motion functions take over.
    set(PARA(1), 'WindowButtonMotionFcn', 'distance(''ButtonMotion'');', ...
          'WindowButtonDownFcn', 'distance(''NextButtonDown'');');

%-----------------------------------------------------
% Subfunction NextButtonDown
%-----------------------------------------------------
function NextButtonDown

global ROW COLUMN PARA LINE_X LINE_Y

pt = get(PARA(9), 'CurrentPoint');
if pt(1,1)>=1 & pt(1,1)<=COLUMN & pt(1,2)>=1 & pt(1,2)<=ROW,
        LINE_X(2)=pt(1,1);
        LINE_Y(2)=pt(1,2);
else return;
end;

    set([PARA(31) PARA(32)], 'XData', LINE_X, ...
          'YData', LINE_Y);
    set(PARA(31), 'UserData', 'Completed');

%-----------------------------------------------------
% Subfunction ButtonMotion
%-----------------------------------------------------
function ButtonMotion

global ROW COLUMN PARA LINE_X LINE_Y


pt= get(PARA(9), 'CurrentPoint');
x = [LINE_X(1) pt(1,1)];
y = [LINE_Y(1) pt(1,2)];

set([PARA(31) PARA(32)], 'XData', x, 'YData', y);
```

Fitting.m

```
function [nans,npv,ndev]=fitting(vect,width)

% Gaussian fit

% Initial parameters

lgth=length(vect);

% Background

high=max(vect);
bg=mean(vect);
level=bg+(high-bg)*0.15;
mask=sparse(vect<level);
nvect=vect.*mask;
if sum(mask)~=0,
   bg=sum(nvect)/sum(mask);
end;
level=bg+(high-bg)*0.15;
mask=sparse(vect<level);
nvect=vect.*mask;
if sum(mask)~=0,
   bg=sum(nvect)/sum(mask);
end;

% Peak

maxX=median(find(vect>(0.85*(high-bg)+bg)));
if isempty(maxX),
  maxX=max(vect);
end;

% Width

if isempty(width)
  width=0;
end;

% Initial

beta0=[high-bg bg maxX width];

% Fit

[nans, r, J]=nlinfit(lgth,vect,'gausbeam',beta0);

% Error

delta=gausbeam(nans,lgth)'-vect;
npv=max(delta)-min(delta);
ndev=std(delta);
```

Gausbeam.m

function y=gausbeam(beta,lgth)

```
% Calculates Gaussian beam 1-D
 X=[1:lgth]';
 X=X-beta(3);
 y=beta(1)*(exp(-8*(X.^2)/beta(4)^2))+beta(2);
```

Graduate.m

function graduate()

```
 % Set axes graduation

 global COLUMN ROW ABSCISSA ORDINATE GRADX GRADY
PXLX PXLY

 gradx=(floor(scaleX(COLUMN)/4)/PXLX);
 if gradx==0
   gradx=(floor(10*scaleX(COLUMN/4))/(10*PXLX));
 end;
 nx=floor(COLUMN/gradx);
 GRADX=[gradx:gradx:nx*gradx];

 grady=(floor(scaleY(ROW)/4)/PXLY);
 if grady==0
                grady=(floor(10*scaleY(ROW/4))/(10*PXLY));
 end;
 ny=floor(ROW/grady);
 GRADY=[ROW-ny*grady:grady:ROW-grady];


 absciss0=sprintf('%4.1f',scaleX(0));
 absciss1=sprintf('%4.1f',scaleX(gradx));
 absciss2=sprintf('%4.1f',scaleX(2*gradx));
 absciss3=sprintf('%4.1f',scaleX(3*gradx));
 absciss4=sprintf('%4.1f',scaleX(4*gradx));
 absciss5=sprintf('%4.1f',scaleX(5*gradx));
 absciss6=sprintf('%4.1f',scaleX(6*gradx));
 absciss7=sprintf('%4.1f',scaleX(7*gradx));


 abscissa=[absciss0;absciss1; absciss2 ;absciss3; absciss4;absciss5
;absciss6; absciss7];
 ABSCISSA=abscissa([1:nx+1],[1:4]);


 ordinat1=sprintf('%4.1f',scaleY(grady));
 ordinat2=sprintf('%4.1f',scaleY(2*grady));
 ordinat3=sprintf('%4.1f',scaleY(3*grady));
 ordinat4=sprintf('%4.1f',scaleY(4*grady));
 ordinat5=sprintf('%4.1f',scaleY(5*grady));
 ordinat6=sprintf('%4.1f',scaleY(6*grady));
 ordinat7=sprintf('%4.1f',scaleY(7*grady));

 ordinate=[ordinat7;ordinat6;ordinat5;ordinat4;ordinat3;ordinat2;ordinat1];
 ORDINATE=ordinate([8-ny:7],[1:4]);
```

Height.m

function ans=height(window)

```
 % Calculate the height of the window

 global PXLX PXLY
 ans=floor(2*sqrt(PXLX/PXLY)*window)-1;

 if (rem(ans,2)==0);
                ans=ans-1;
 end;
```

Initaxes.m

function initaxes()

```
% Initialize AXES

global WINDOW DIAGMAX ROW COLUMN GRADX GRADY
GRADY ABSCISSA ORDINATE PARA

h=PARA(1);

%=======================================
% VIEW axes
%=======================================


 h_view_axis = axes(                              ...
    'Parent',h,                                ...
    'Units', 'normalized',                     ...
    'Position', [-0.05 0.24 0.592 0.592],      ...
    'BusyAction','Queue',                      ...
    'DataAspectRatioMode','manual',            ...
    'DataAspectRatio',[1 ROW/COLUMN 1],        ...
    'Box','on',                                ...
    'DrawMode','fast',                         ...
    'Tag', 'ViewAxes',                         ...
    'NextPlot','replace');

%=======================================
% ANALYSIS axes
%=======================================

 h_analysis_axis = axes(                          ...
    'Parent',h,                                ...
    'Units', 'normalized',                     ...
    'Position', [0.37 0.24 0.74 0.74],         ...
    'BusyAction','Queue',                      ...
    'DataAspectRatioMode','manual',            ...
    'DataAspectRatio',[1 ROW/COLUMN 1],        ...
    'DrawMode','fast',                         ...
    'Tag', 'AnalysisAxes',                     ...
    'NextPlot','replace',                      ...
    'Box', 'on');


%====================================
% Colorbar
%====================================
```

```
h_color_axis = axes(                                    ...
   'Parent',h,                                  ...
   'Units', 'normalized',                       ...
   'Position', [0.027 0.24 0.014 0.592],                ...
   'Box', 'on');

% Create color stripe
%====================

n = size(colormap,1);
h_stripe=image([0 1],caxis,[1:n]','Tag','ColorStripe');
set(h_color_axis,'YDir','normal',                       ...
   'XTick',[],                                  ...
   'YGrid','on',                                ...
   'FontSize',11,                               ...
   'FontWeight','bold',                         ...
   'DrawMode','fast',                           ...
   'Tag', 'ColorAxes');

PARA(8)=findobj('Tag','ViewAxes');
PARA(9)=findobj('Tag','AnalysisAxes');

% Set ViewAxes
%==============

newview;

% Set AnalysisAxes
%==============

newcont;
```

Initwind.m

```
function initwind()

% Initialize WINDOW slider

global WINDOW DIAGMAX PARA

h=findobj('Tag','BEAMDIAG');

w1=round(WINDOW(1)+(width(WINDOW(3))+1)/2);
w2=511-round(WINDOW(2)+(height(WINDOW(3))+1)/2);
w3=round(WINDOW(3));

%====================================
% WINDOW frame
%====================================

uicontrol(                                      ...
                'Parent',h,                             ...
 'Style','frame',                               ...
                'Units','normalized',                   ...
 'Position',[0.454 0.01 0.229 0.2]);

uicontrol(                                      ...
                'Parent',h,                             ...
 'Style','text',                                ...
                'Units','normalized',                   ...
                'Position',[0.511 0.173 0.11 0.03],             ...
                'String','WINDOW',                      ...
 'FontSize',13,                                 ...
 'FontWeight','bold');

%====================================
% Initialize OK button
%====================================

h_ok= uicontrol(                                        ...
   'Parent',h,                                  ...
   'Style', 'pushbutton',                       ...
   'Units', 'normalized',                       ...
   'String', 'OK',                              ...
   'FontSize',12,                               ...
   'FontWeight','bold',                         ...
   'Tag','okwind',                              ...
   'Position', [0.634 0.144 0.0355 0.032],              ...
   'Callback', 'setwind');
set(h_ok,'Enable','off');

%====================================
% Initialize slider label up/down
%====================================

uicontrol(                                      ...
   'Parent',h,                                  ...
   'Style', 'text',                             ...
   'Units', 'normalized',                       ...
   'String', 'Up',                              ...
   'FontSize',12,                               ...
   'FontWeight','bold',                         ...
   'Position', [0.46 0.149 0.03 0.03],                  ...
   'HorizontalAlignment', 'center',                     ...
   'Foreground', 'black');

uicontrol(                                      ...
   'Parent',h,                                  ...
   'Style', 'text',                             ...
   'Units', 'normalized',                       ...
   'String', 'Down',                            ...
   'FontSize',12,                               ...
   'FontWeight','bold',                         ...
   'Position', [0.46 0.0115 0.045 0.03],                ...
   'HorizontalAlignment', 'center',                     ...
   'Foreground', 'black');

%====================================
% Initialize editable text up/down
%====================================

uicontrol(                                      ...
   'Parent',h,                                  ...
   'Style', 'edit',                             ...
   'Units', 'normalized',                       ...
   'String', num2str(w2),                       ...
   'FontSize',11,                               ...
   'FontWeight','bold',                         ...
   'Position', [0.485 0.116 0.042 0.03],                ...
   'Interruptible', 'off',                      ...
   'Foreground', 'black',                       ...
   'Callback', 'editup',                        ...
   'Tag', 'editup',                             ...
   'HorizontalAlignment', 'center');

%====================================
% Initialize slider up/down
%====================================

 minV=min([round((height(WINDOW(3))+1)/2),511-
round((height(WINDOW(3))+1)/2)]);
 maxV=max([round((height(WINDOW(3))+1)/2),511-
round((height(WINDOW(3))+1)/2)]);

uicontrol(                                      ...
   'Parent',h,                                  ...
   'Style', 'slider',                           ...
   'Units', 'normalized',                       ...
   'Position',[0.465 0.042 0.018 0.104],                ...
```

```
    'Min', minV,                                    ...
    'Max', maxV,                                     ...
    'Value', w2,                                    ...
    'Tag','sliderup',                                ...
    'Interruptible', 'off',                         ...
    'Callback', 'sliderup');

%====================================
% Initialize slider label right/left
%====================================

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'text',                                ...
    'Units', 'normalized',                           ...
    'String', 'Left',                               ...
    'FontSize',12,                                   ...
    'FontWeight','bold',                             ...
    'Position', [0.529 0.084 0.04 0.03],                    ...
    'HorizontalAlignment', 'center',                        ...
    'Foreground', 'black');

 uicontrol(                                         ...
    'Parent',h,                                     ...
    'Style', 'text',                                ...
    'Units', 'normalized',                           ...
    'String', 'Right',                              ...
    'FontSize',12,                                   ...
    'FontWeight','bold',                             ...
    'Position', [0.64 0.084 0.041 0.03],                    ...
    'HorizontalAlignment', 'center',                        ...
    'Foreground', 'black');

%====================================
% Initialize editable text right/left
%====================================

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'edit',                                ...
    'Units', 'normalized',                           ...
    'String', num2str(w1),                           ...
    'FontSize',11,                                  ...
    'FontWeight','bold',                             ...
    'Position', [0.595 0.085 0.042 0.03],                   ...
    'Interruptible', 'off',                         ...
    'Foreground', 'black',                           ...
    'Callback', 'editleft',                         ...
    'Tag', 'editleft',                              ...
    'HorizontalAlignment', 'center');

%====================================
% Initialize slider right/left
%====================================

 minV=min([round((width(WINDOW(3))+1)/2),511-
round((width(WINDOW(3))+1)/2)]);
 maxV=max([round((width(WINDOW(3))+1)/2),511-
round((width(WINDOW(3))+1)/2)]);

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'slider',                              ...
    'Units', 'normalized',                           ...
    'Position',[0.532 0.115 0.14 0.021],                    ...
    'Min', minV,                                    ...
    'Max', maxV,                                     ...
    'Value', w1,                                    ...
    'Tag','sliderle',                               ...
    'Interruptible', 'off',                         ...
    'Callback', 'sliderle');
```

```
%====================================
% Initialize SIZE slider label
%====================================

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'text',                                ...
    'Units', 'normalized',                           ...
    'String', 'Part',                               ...
    'FontSize',12,                                   ...
    'FontWeight','bold',                             ...
    'Position', [0.529 0.0115 0.035 0.03],                  ...
    'HorizontalAlignment', 'center',                        ...
    'Foreground', 'black');

 uicontrol(                                         ...
    'Parent',h,                                     ...
    'Style', 'text',                                ...
    'Units', 'normalized',                           ...
    'String', 'Full',                               ...
    'FontSize',12,                                   ...
    'FontWeight','bold',                             ...
    'Position', [0.645 0.0115 0.035 0.03],                  ...
    'HorizontalAlignment', 'center',                        ...
    'Foreground', 'black');

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'text',                                ...
    'Units', 'normalized',                           ...
    'String', 'Size',                               ...
    'FontSize',9,                                    ...
    'FontWeight','bold',                             ...
    'Position', [0.567 0.0118 0.03 0.02],                   ...
    'HorizontalAlignment', 'center',                        ...
    'Foreground', 'black');

%====================================
% Initialize SIZE editable text
%====================================

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'edit',                                ...
    'Units', 'normalized',                           ...
    'String', num2str(w3),                           ...
    'FontSize',11,                                  ...
    'FontWeight','bold',                             ...
    'Position', [0.595 0.0115 0.042 0.03],                  ...
    'Interruptible', 'off',                         ...
    'Foreground', 'black',                           ...
    'Callback', 'editsize',                         ...
    'Tag', 'editsize',                              ...
    'HorizontalAlignment', 'center');

%====================================
% Initialize SIZE slider
%====================================

uicontrol(                                          ...
    'Parent',h,                                     ...
    'Style', 'slider',                              ...
    'Units', 'normalized',                           ...
    'Position',[0.532 0.042 0.14 0.021],                    ...
    'Min', 70,                                      ...
    'Max', DIAGMAX,                                  ...
    'Value', w3,                                    ...
    'Tag','slidersi',                               ...
    'Interruptible', 'off',                         ...
    'Callback', 'slidersi');

 PARA(7)=findobj('Tag','okwind');
```

Menubeam.m

```
function menubeam()

  global BEAM_DAT PARA

  h=PARA(1);
  BEAM_DAT=[2 1 1 1 1 1 0 2 1 1 1 0];

  %====================================
  % Surface (1) or Line (0) 3D_Plot
  %====================================

  BEAM_DAT(12)=0;

  %====================================
  % View menu
  %====================================

  h_view = uimenu('Parent',h,'Label','&View');
  h_video = uimenu('Parent',h_view,'Label','&Video','Tag','Camera',
...
       'CallBack','setcam');
  h_normalized =
uimenu('Parent',h_view,'Label','&Normalized','Tag','Normal',  ...
       'CallBack','setnorm');
  h_off1 = uimenu('Parent',h_view,'Label','&Off','Tag','Off1',
...
       'CallBack','setoff1');
  set(h_normalized,'Checked','on');
  BEAM_DAT(1)=2;


  %====================================
  % Analysis menu
  %====================================

  h_analysis = uimenu('Parent',h,'Label','&Analysis');
  h_contour =
uimenu('Parent',h_analysis,'Label','&Contour','Tag','Contour',   ...
       'CallBack','setcon');
  h_3d = uimenu('Parent',h_analysis,'Label','3&D','Tag','3D',
...
       'CallBack','set3d');
  h_off2 = uimenu('Parent',h_analysis,'Label','&Off','Tag','Off2',
...
       'CallBack','setoff2');
  set(h_contour,'Checked','on');
  BEAM_DAT(2)=1;
  h_barycentre =
uimenu('Parent',h_analysis,'Label','&Barycentre','Tag','Bary1',...
       'CallBack','setbary1','Separator','on');
  h_max = uimenu('Parent',h_analysis,'Label','&Maximum','Tag','Max1',
...
       'CallBack','setmax1');
  h_off3 = uimenu('Parent',h_analysis,'Label','Of&f','Tag','Off3',
...
       'CallBack','setoff3');
  set(h_barycentre,'Checked','on');
  BEAM_DAT(3)=1;


  %====================================
  % Profile menu
  %====================================

  h_profile = uimenu('Parent',h,'Label','&Profile');
  h_bary2 =
uimenu('Parent',h_profile,'Label','&Barycentre','Tag','Bary2',     ...
       'CallBack','setbary2');
  h_max2 = uimenu('Parent',h_profile,'Label','&Maximum','Tag','Max2',
...
       'CallBack','setmax2');
  h_pers = uimenu('Parent',h_profile,'Label','&Personal','Tag','Pers',        ...
       'CallBack','setpers');
  h_off4 = uimenu('Parent',h_profile,'Label','&Off','Tag','Off4',             ...
       'CallBack','setoff4');
  set(h_bary2,'Checked','on');
  BEAM_DAT(4)=1;
  h_integrated =
uimenu('Parent',h_profile,'Label','&Integrated','Tag','Integr',...
     'CallBack','setinteg','Separator','on');
  set(h_integrated,'Checked','on');
  BEAM_DAT(5)=1;

  %====================================
  % Options menu
  %====================================

  h_options = uimenu('Parent',h,'Label','&Options');
  h_bary3 = uimenu('Parent',h_options,'Label','&Barycentre','Tag','Bary3');
  h_bary4 = uimenu('Parent',h_bary3,'Label','&Manual','Tag','Bary4',
...
                       'CallBack','setbary4');
  h_bary5 = uimenu('Parent',h_bary3,'Label','&Automatic','Tag','Bary5',
...
                       'CallBack','setbary5');
  h_width1 =
uimenu('Parent',h_options,'Label','&Widths','Tag','Width1','Separator','on');
  h_width2 = uimenu('Parent',h_width1,'Label','&Profiles','Tag','Width2',
...
                       'CallBack','setwid2');
  h_width3 = uimenu('Parent',h_width1,'Label','&Integrated','Tag','Width3',
...
     'CallBack','setwid3');
  h_background =
uimenu('Parent',h_options,'Label','Ba&ckground','Tag','Backg',...
       'CallBack','setback','Separator','on');
  h_reference = uimenu('Parent',h_options,'Label','&Mark
Reference','Tag','Refer',...
       'CallBack','setref','Separator','on');
  h_distance = uimenu('Parent',h_options,'Label','&Length','Tag','Length',
...
       'CallBack','distance','Separator','on');
  h_fit = uimenu('Parent',h_options,'Label','&Fit','Tag','Fit',               ...
     'Separator','on');
  h_fit1 = uimenu('Parent',h_fit,'Label','P&rofiles','Tag','Fit1',      ...
                       'CallBack','setfit1');
  h_fit2 = uimenu('Parent',h_fit,'Label','I&ntegrated','Tag','Fit2',    ...
     'CallBack','setfit2');
  h_fit3 = uimenu('Parent',h_fit,'Label','&Off','Tag','Fit3',    ...
     'CallBack','setfit3');
  h_print = uimenu('Parent',h_options,'Label','&Print','Tag','Print',          ...
     'Separator','on',                                                  ...
     'CallBack','printwin');
  h_save = uimenu('Parent',h_options,'Label','&Save','Tag','Zoom',
...
       'CallBack','beamctf("SaveBEAM")','Separator','on');
    set(h_bary2,'Checked','on');
  set(h_bary5,'Checked','on');
  set(h_width3,'Checked','on');
  set(h_fit2,'Checked','on');
  set(h_reference,'Checked','on');
  set(h_background,'Checked','on');
  BEAM_DAT(6)=1;
  BEAM_DAT(7)=0;
  BEAM_DAT(8)=2;
  BEAM_DAT(9)=1;
  BEAM_DAT(10)=1;
  BEAM_DAT(11)=1;

  %====================================
  % Help menu
  %====================================
```

```
  h_help = uimenu('Parent',h,'Label','&Help');
  h_help1 = uimenu('Parent',h_help,'Label','&Beamdiag','Tag','Help1',
...
      'CallBack','help1');
  h_help2 = uimenu('Parent',h_help,'Label','&View','Tag','Help2',
...
      'CallBack','help2');
  h_help3 = uimenu('Parent',h_help,'Label','&Analysis','Tag','Help3',
...
      'CallBack','help3');
  h_help4 = uimenu('Parent',h_help,'Label','&Profile','Tag','Help4',
...
      'CallBack','help4');
  h_help5 = uimenu('Parent',h_help,'Label','&Options','Tag','Help5',
...
      'CallBack','help5');
```

```
%====================================
% Run menu
%====================================

h_empty = uimenu('Parent',h,'Label',' ');
h_play2 = uimenu('Parent',h,'Label','Pla&y','Tag','Play2',                ...
   'Interruptible','on',                                                  ...
   'Callback','beamctf("DisplayBEAM")');
h_stop2 = uimenu('Parent',h,'Label','&Stop','Tag','Stop2',                ...
   'Callback','set(findobj("Tag","Play"),"Userdata",0)');
h_close2 = uimenu('Parent',h,'Label','&Close','Tag','Close2',             ...
   'Callback','beamctf("CloseBEAM")');

PARA(2)=findobj('Tag','Play2');
PARA(3)=findobj('Tag','Close2');
```

Newcont.m

```
function newcont()

% Initialize Axes for new Contour

global ROW COLUMN PARA GRADX GRADY CONTOURTH
ABSCISSA ORDINATE

AnalysisAxes=PARA(9);

% Set Analysis Axes

axes(AnalysisAxes);
set(AnalysisAxes,'NextPlot','replace');

plot([1 5*COLUMN/4],[0 0],'black');

set(AnalysisAxes,                                  ...
                  'Units', 'normalized',                                ...
   'Position', [0.37 0.24 0.74 0.74],                 ...
   'TickDir','in',                                    ...
   'XLim',[1 5*COLUMN/4],                             ...
   'XTick', [1 GRADX],                                ...
   'XTickLabel',ABSCISSA,                             ...
   'FontSize',11,                                     ...
   'FontWeight','bold',                               ...
   'XGrid','on',                                      ...
   'YDir','reverse',                                  ...
   'YLim',[-ROW/4 ROW],                               ...
   'YTick',GRADY,                                     ...
   'YTickLabel',ORDINATE,                             ...
   'YGrid','on',                                      ...
   'DataAspectRatioMode','manual',                    ...
   'DataAspectRatio',[1 ROW/COLUMN 1],                ...
                     'Tag','AnalysisAxes',
...
   'NextPlot','add');

plot([COLUMN COLUMN],[-ROW/4 ROW],'black');
```

Newview.m

```
function newview()

% Initialize Axes for new View

global ROW COLUMN PARA GRADX GRADY

mapfig=get(PARA(1),'Colormap');

ViewAxes=PARA(8);

% Set ViewAxes

axes(ViewAxes);
h_pict=imshow(zeros(ROW,COLUMN));
colormap(mapfig);
set(ViewAxes,                                      ...
   'TickDir','in',                                 ...
   'XTick',GRADX,                                  ...
   'XTickLabel','',                                ...
   'XColor','w',                                   ...
   'XGrid','on',                                   ...
   'YTick',GRADY,                                  ...
   'YTickLabel','',                                ...
   'YColor','w',                                   ...
   'YGrid','on',                                   ...
   'DataAspectRatioMode','manual',                 ...
   'DataAspectRatio',[1 ROW/COLUMN 1],             ...
   'Tag','ViewAxes',                               ...
   'Visible','on');

PARA(30)=h_pict;
```

Nlinfit.m

```
function [beta,r,J] = nlinfit(X,y,model,beta0)
%NLINFIT Nonlinear least-squares data fitting by the Gauss-Newton
method.
%   NLINFIT(X,Y,'MODEL',BETA0) finds the coefficients of the
nonlinear
%   function described in MODEL. MODEL is a user supplied function
having
%   the form y = f(beta,x). That is MODEL returns the predicted values of
y
```

```
%   given initial parameter estimates, beta, and the independent variable,
X.
%   [BETA,R,J] = NLINFIT(X,Y,'MODEL',BETA0) returns the fitted
coefficients
%   BETA the residuals, R, and the Jacobian, J, for use with NLINTOOL
to
%   produce error estimates on predictions.

%   B.A. Jones 12-06-94.
%   Copyright (c) 1993-96 by The MathWorks, Inc.
%   $Revision: 2.7 $  $Date: 1996/09/29 17:33:27 $

n = length(y);
if min(size(y)) ~= 1
  error('Requires a vector second input argument.');
end
y = y(:);

p = length(beta0);
beta0 = beta0(:);

J = zeros(n,p);
beta = beta0;
betanew = beta + 1;
maxiter = 100;
iter = 0;
betatol = 1.0E-4;
rtol = 1.0E-4;
sse = 1;
sseold = sse;

while (norm((betanew-beta)./(beta+sqrt(eps))) > betatol | abs(sseold-
sse)/(sse+sqrt(eps)) > rtol) & iter < maxiter
  if iter > 0,
    beta = betanew;
  end

  iter = iter + 1;
  evalstr = [model,'(beta,X)'];
  yfit = eval(evalstr);
  r = y - yfit;
  sseold = r'*r;

  for k = 1:p,
    delta = zeros(size(beta));
    delta(k) = sqrt(eps)*beta(k);
    evalstr1 = [model,'(beta+delta,X)'];
    yplus = eval(evalstr1);
    J(:,k) = (yplus - yfit)/(sqrt(eps)*beta(k));
  end

  Jplus = [J;1.0E-2*eye(p)];
  rplus = [r;zeros(p,1)];

  % Levenberg-Marquardt type adjustment
  % Gauss-Newton step -> J\r
  % LM step -> inv(J'*J+constant*eye(p))*J'*r
  step = Jplus\rplus;

  betanew = beta + step;
  evalstrnew = [model,'(betanew,X)'];
  yfitnew = eval(evalstrnew);
  rnew = y - yfitnew;
  sse = rnew'*rnew;
  iter1(iter) = 0;
  while sse > sseold & iter1(iter) < 12
    step = step/sqrt(10);
    betanew = beta + step;
    evalstrnew = [model,'(betanew,X)'];
    yfitnew = eval(evalstrnew);
    rnew = y - yfitnew;
    sse = rnew'*rnew;
    iter1(iter) = iter1(iter) + 1;
  end
end
```

Noise.m

```
function ans=noise(mat)

% Calculate the background of the normalized matrice

bg=mean(mean(mat));
level=bg+(1-bg)*0.2;
mask=sparse(mat<level);
nmat=mat.*mask;
bg=sum(nmat)/sum(mask);
level=bg+(1-bg)*0.2;
mask=sparse(mat<level);
nmat=mat.*mask;
bg=sum(nmat)/sum(mask);
maskl=sparse(mat<bg)*bg;
maskh=sparse(mat>bg).*mat;
mat=full(maskl+maskh);
ans=(mat-bg)/(1-bg);
```

ScaleX.m

```
function r=scaleX(x);

                % Horizontal dimension
                                                    r=PXLX*x;
 global PXLX
```

ScaleY.m

```
function r=scaleY(y);

                % Vertical dimension
                                                    r=PXLY*y;
 global PXLY
```

Setframe.m

function setframe()

```
global CONTOURTH BARTH PARA

h=PARA(1);

%====================================
% LEVEL frame
%====================================

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','frame',                            ...
    'Units','normalized',                   ...
'Position',[0.047 0.847 0.34 0.138]);

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.114 0.951 0.2 0.033],     ...
'String','VIDEO LEVELS ( % )',              ...
'FontSize',13,                              ...
'FontWeight','bold',                        ...
'ForegroundColor','b');

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.17 0.916 0.04 0.03],      ...
    'String','High',                        ...
    'HorizontalAlignment','left',           ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'ForegroundColor','b');

h_high_label=uicontrol(                     ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.22 0.916 0.035 0.03],     ...
    'HorizontalAlignment','left',           ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'Tag','highlabel',                          ...
'BackgroundColor','w');

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.055 0.916 0.04 0.03],     ...
    'String','Low',                         ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'HorizontalAlignment','left',               ...
    'ForegroundColor','b');

h_low_label=uicontrol(                      ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.123 0.916 0.035 0.03],    ...
    'String','',                            ...
    'HorizontalAlignment','left',           ...
'FontSize',11,                              ...
'FontWeight','bold',                        ...
'Tag','lowlabel',                           ...
'BackgroundColor','w');

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.055 0.862 0.065 0.03],    ...
    'String','Average',                     ...
'FontSize',11,                              ...
'FontWeight','bold',                        ...
'HorizontalAlignment','left',               ...
'ForegroundColor','b');

h_avg_label=uicontrol(                      ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.123 0.862 0.035 0.03],    ...
    'HorizontalAlignment','left',           ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'Tag','avglabel',                           ...
'BackgroundColor','w');

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.17 0.862 0.04 0.03],      ...
    'String','RMS',                         ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'HorizontalAlignment','left',               ...
    'ForegroundColor','b');

h_rms_label=uicontrol(                      ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.22 0.862 0.035 0.03],     ...
    'String','',                            ...
    'HorizontalAlignment','left',           ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'Tag','rmslabel',                           ...
'BackgroundColor','w');

h_saturation=uicontrol(                     ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.265 0.86 0.11 0.06],      ...
    'HorizontalAlignment','center',         ...
'Tag','saturationlabel',                    ...
'BackgroundColor','r',                      ...
'Visible','off');

uicontrol(                                  ...
    'Parent',h,                             ...
'Style','text',                             ...
    'Units','normalized',                   ...
    'Position',[0.265 0.921 0.111 0.03],    ...
    'String','SATURATION',                  ...
    'FontSize',11,                          ...
'FontWeight','bold',                        ...
'HorizontalAlignment','center',             ...
'ForegroundColor','b');

%====================================
% LABEL frame
%====================================

uicontrol(                                  ...
    'Parent',h,                             ...
```

```
 'Style','frame',                                    ...
     'Units','normalized',                           ...
 'Position',[0.893 0.835 0.097 0.142]);

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.894 0.945 0.094 0.028],           ...
 'String',['CntTh : ' sprintf('%3.0f',100*CONTOURTH(1)) '%'],...
 'FontSize',10,                                      ...
 'FontWeight','bold',                                ...
     'ForegroundColor',[0 0.2 1]);

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.894 0.919 0.094 0.028],           ...
 'String',['CntTh : ' sprintf('%3.0f',100*CONTOURTH(2)) '%'],...
 'FontSize',10,                                      ...
 'FontWeight','bold',                                ...
     'ForegroundColor',[0.4 1 0.6]);

h_label_contourth=uicontrol(                         ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.894 0.893 0.094 0.028],           ...
     'Tag','ContourTh',                              ...
     'String',['CntTh : ' sprintf('%3.0f',100*CONTOURTH(3)) '%'],...
     'FontSize',10,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor','r');

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.894 0.867 0.094 0.028],           ...
     'String','Profile',                             ...
     'FontSize',10,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor',[0 0 0.8]);

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.894 0.84 0.094 0.029],            ...
     'String','Integrated',                          ...
     'FontSize',10,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor',[1 0.5 0]);


%====================================
% POSITION frame
%====================================

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','frame',                                    ...
     'Units','normalized',                           ...
 'Position',[0.01 0.01 0.21 0.2]);

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.032 0.173 0.17 0.035],            ...
     'String','POSITION ( mm )',                     ...
     'FontSize',13,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.13 0.137 0.016 0.03],             ...
     'String','X',                                   ...
     'FontSize',13,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.182 0.137 0.016 0.03],            ...
     'String','Y',                                   ...
     'FontSize',13,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.017 0.108 0.045 0.027],           ...
     'String','Mark',                                ...
     'HorizontalAlignment','left',                   ...
     'FontSize',11,                                  ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

h_mark_x=uicontrol(                                  ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.117 0.108 0.042 0.027],           ...
     'HorizontalAlignment','left',                   ...
     'Tag','markx',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_mark_y=uicontrol(                                  ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.17 0.108 0.042 0.027],            ...
     'HorizontalAlignment','left',                   ...
     'Tag','marky',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

uicontrol(                                           ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.017 0.078 0.081 0.027],           ...
     'String','Movement',                            ...
     'FontSize',11,                                  ...
 'FontWeight','bold',                                ...
 'HorizontalAlignment','left',                       ...
     'ForegroundColor','b');

h_move_x=uicontrol(                                  ...
     'Parent',h,                                     ...
 'Style','text',                                     ...
     'Units','normalized',                           ...
     'Position',[0.117 0.078 0.042 0.027],           ...
     'HorizontalAlignment','left',                   ...
     'Tag','movex',                                  ...
 'FontSize',11,                                      ...
```

```
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_move_y=uicontrol(                                  ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.17 0.078 0.042 0.027],           ...
      'HorizontalAlignment','left',                  ...
      'Tag','movey',                                 ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

uicontrol(                                           ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.017 0.048 0.06 0.027],           ...
      'String','FWHM',                               ...
      'FontSize',11,                                 ...
 'FontWeight','bold',                                ...
 'HorizontalAlignment','left',                       ...
      'ForegroundColor','b');

h_FWHH_x=uicontrol(                                  ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.117 0.048 0.042 0.027],          ...
      'HorizontalAlignment','left',                  ...
      'Tag','fwhhx',                                 ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_FWHH_y=uicontrol(                                  ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.17 0.048 0.042 0.027],           ...
      'HorizontalAlignment','left',                  ...
      'Tag','fwhhy',                                 ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

uicontrol(                                           ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.017 0.018 0.077 0.027],          ...
      'String','Diameter',                           ...
      'HorizontalAlignment','left',                  ...
      'FontSize',11,                                 ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

h_waist_x=uicontrol(                                 ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.117 0.018 0.042 0.027],          ...
      'HorizontalAlignment','left',                  ...
      'Tag','waistx',                                ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_waist_y=uicontrol(                                 ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.17 0.018 0.042 0.027],           ...
      'HorizontalAlignment','left',                  ...
      'Tag','waisty',                                ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

%====================================
% Initialize INIT button
%====================================

 h_init= uicontrol(                                  ...
    'Parent',h,                                      ...
    'Style', 'pushbutton',                           ...
    'Units', 'normalized',                           ...
    'String', 'Init',                                ...
    'FontSize',10,                                   ...
    'FontWeight','bold',                             ...
    'Tag','initref',                                 ...
    'Position', [0.06 0.106 0.0355 0.032],           ...
    'Callback', 'initref');


 %====================================
 % FIT frame
 %====================================

 uicontrol(                                          ...
      'Parent',h,                                    ...
 'Style','frame',                                    ...
      'Units','normalized',                          ...
 'Position',[0.232 0.05 0.21 0.16]);

uicontrol(                                           ...
      'Parent',h,                                    ...
 'Style','frame',                                    ...
      'Units','normalized',                          ...
 'Position',[0.232 0.01 0.21 0.037]);

 uicontrol(                                          ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.309 0.173 0.06 0.035],           ...
      'String','FIT',                                ...
      'FontSize',13,                                 ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

 uicontrol(                                          ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.352 0.145 0.016 0.03],           ...
      'String','X',                                  ...
      'FontSize',13,                                 ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

 uicontrol(                                          ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.402 0.145 0.016 0.03],           ...
      'String','Y',                                  ...
      'FontSize',13,                                 ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

 uicontrol(                                          ...
      'Parent',h,                                    ...
 'Style','text',                                     ...
      'Units','normalized',                          ...
      'Position',[0.239 0.115 0.1 0.027],            ...
      'String','FWHM (mm)',                          ...
```

```
    'HorizontalAlignment','left',                    ...
    'FontSize',11,                                   ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

h_fmark_x=uicontrol(                                 ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.339 0.115 0.042 0.027],            ...
    'HorizontalAlignment','left',                    ...
    'Tag','fmarkx',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_fmark_y=uicontrol(                                 ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.39 0.115 0.042 0.027],             ...
    'HorizontalAlignment','left',                    ...
    'Tag','fmarky',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

uicontrol(                                           ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.239 0.086 0.1 0.027],              ...
    'String','Pk-Vly (%)',                           ...
    'FontSize',11,                                   ...
 'FontWeight','bold',                                ...
 'HorizontalAlignment','left',                       ...
    'ForegroundColor','b');

h_fmove_x=uicontrol(                                 ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.339 0.086 0.042 0.027],            ...
    'HorizontalAlignment','left',                    ...
    'Tag','fmovex',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_fmove_y=uicontrol(                                 ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.39 0.086 0.042 0.027],             ...
    'HorizontalAlignment','left',                    ...
    'Tag','fmovey',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

uicontrol(                                           ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.239 0.055 0.1 0.027],              ...
    'String','Std Dev (%)',                          ...
    'FontSize',11,                                   ...
 'FontWeight','bold',                                ...
 'HorizontalAlignment','left',                       ...
    'ForegroundColor','b');

h_fFWHH_x=uicontrol(                                 ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.339 0.055 0.042 0.027],            ...
    'HorizontalAlignment','left',                    ...
    'Tag','ffwhhx',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_fFWHH_y=uicontrol(                                 ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.39 0.055 0.042 0.027],             ...
    'HorizontalAlignment','left',                    ...
    'Tag','ffwhhy',                                  ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

uicontrol(                                           ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.239 0.014 0.077 0.029],            ...
    'String','Length',                               ...
    'HorizontalAlignment','left',                    ...
    'FontSize',11,                                   ...
 'FontWeight','bold',                                ...
 'ForegroundColor','b');

h_fwaist_x=uicontrol(                                ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.339 0.014 0.042 0.027],            ...
    'HorizontalAlignment','left',                    ...
    'Tag','fwaistx',                                 ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

h_fwaist_y=uicontrol(                                ...
    'Parent',h,                                      ...
 'Style','text',                                     ...
    'Units','normalized',                            ...
    'Position',[0.39 0.014 0.042 0.027],             ...
    'HorizontalAlignment','left',                    ...
    'Tag','fwaisty',                                 ...
 'FontSize',11,                                      ...
 'FontWeight','bold',                                ...
 'BackgroundColor','w');

%====================================
% CONTROL frame
%====================================

uicontrol(                                           ...
    'Parent',h,                                      ...
 'Style','frame',                                    ...
    'Units','normalized',                            ...
 'Position',[0.695 0.01 0.3 0.2]);

%====================================
% Initialize BARYCENTRE sliders
%====================================

%====================================
% Initialize slider label
%====================================

uicontrol(                                           ...
   'Parent',h,                                       ...
   'Style', 'text',                                  ...
   'Units', 'normalized',                            ...
```

```
    'String', 'BarLow',                                 ...
    'FontSize',12,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.775 0.17 0.07 0.03],                 ...
    'HorizontalAlignment', 'center',                    ...
    'Foreground', 'black');

 uicontrol(                                             ...
    'Parent',h,                                         ...
    'Style', 'text',                                    ...
    'Units', 'normalized',                              ...
    'String', 'BarHigh',                                ...
    'FontSize',12,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.855 0.17 0.07 0.03],                 ...
    'HorizontalAlignment', 'center',                    ...
    'Foreground', 'black');

%====================================
% Initialize editable text
%====================================

h_edit_barl = uicontrol(                                ...
    'Parent',h,                                         ...
    'Style', 'edit',                                    ...
    'Units', 'normalized',                              ...
    'String', num2str(100*BARTH(1)),                    ...
    'FontSize',11,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.799 0.13 0.042 0.03],                ...
    'Interruptible', 'off',                             ...
    'Foreground',[0.5 0.5 0.5],                         ...
    'Callback', 'editbarl',                             ...
    'Tag', 'editbarl',                                  ...
    'HorizontalAlignment', 'center');

 h_text_barl = uicontrol(                               ...
    'Parent',h,                                         ...
    'Style', 'text',                                    ...
    'Units', 'normalized',                              ...
    'FontSize',11,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.799 0.09 0.042 0.03],                ...
    'Tag', 'textbarl',                                  ...
    'BackgroundColor','w',                              ...
    'HorizontalAlignment', 'center');

uicontrol(                                              ...
    'Parent',h,                                         ...
    'Style', 'edit',                                    ...
    'Units', 'normalized',                              ...
    'String', num2str(100*BARTH(2)),                    ...
    'FontSize',11,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.877 0.13 0.042 0.03],                ...
    'Interruptible', 'off',                             ...
    'Foreground','black',                               ...
    'Callback', 'editbarh',                             ...
    'Tag', 'editbarh',                                  ...
    'HorizontalAlignment', 'center');


%====================================
% Initialize slider
%====================================

h_slider_contour = uicontrol(                           ...
    'Parent',h,                                         ...
    'Style', 'slider',                                  ...
    'Units', 'normalized',                              ...
    'Position',[0.779 0.03 0.018 0.135],                ...
    'Min', 0,                                           ...
    'Max', 100*BARTH(2),                                ...
    'Value', 100*BARTH(1),                              ...
    'Tag','sliderbarl',                                 ...
    'Interruptible', 'off',                             ...
    'Callback', 'sliderbl');

uicontrol(                                              ...
    'Parent',h,                                         ...
    'Style', 'slider',                                  ...
    'Units', 'normalized',                              ...
    'Position',[0.857 0.03 0.018 0.135],                ...
    'Min', 100*BARTH(1),                                ...
    'Max', 100,                                         ...
    'Value', 100*BARTH(2),                              ...
    'Tag','sliderbarh',                                 ...
    'Interruptible', 'off',                             ...
    'Callback', 'sliderbh');


%====================================
% Initialize CONTOUR slider
%====================================

%====================================
% Initialize slider label
%====================================

uicontrol(                                              ...
    'Parent',h,                                         ...
    'Style', 'text',                                    ...
    'Units', 'normalized',                              ...
    'String', 'CntTh',                                  ...
    'FontSize',12,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.7 0.17 0.06 0.03],                   ...
    'HorizontalAlignment', 'center',                    ...
    'Foreground', 'black');

%====================================
% Initialize editable text
%====================================

h_edit_contour = uicontrol(                             ...
    'Parent',h,                                         ...
    'Style', 'edit',                                    ...
    'Units', 'normalized',                              ...
    'String', num2str(round(100*CONTOURTH(3))),         ...
    'FontSize',11,                                      ...
    'FontWeight','bold',                                ...
    'Position', [0.725 0.13 0.042 0.03],                ...
    'Interruptible', 'off',                             ...
    'Callback', 'editcont',                             ...
    'Tag', 'editcontour',                               ...
    'HorizontalAlignment', 'center');


%====================================
% Initialize slider
%====================================
h_slider_contour = uicontrol(                           ...
    'Parent',h,                                         ...
    'Style', 'slider',                                  ...
    'Units', 'normalized',                              ...
    'Position',[0.705 0.03 0.018 0.135],                ...
    'Min', 55,                                          ...
    'Max', 99,                                          ...
    'Value', 100*CONTOURTH(3),                          ...
    'Tag','slidercont',                                 ...
    'Interruptible', 'off',                             ...
    'Callback', 'sliderco');




%====================================
% Initialize PLAY button
```

```
%====================================

h_play = uicontrol(                                    ...
   'Parent',h,                                    ...
   'Style', 'pushbutton',                         ...
   'Units', 'normalized',                         ...
   'String', 'Play',                              ...
   'FontSize',12,                                 ...
   'FontWeight','bold',                           ...
   'Position', [0.93 0.1495 0.055 0.042],                  ...
   'Interruptible','on',                          ...
   'Tag','Play',                                  ...
   'Callback','beamctf(''DisplayBEAM'')');

%====================================
% Initialize STOP button
%====================================

h_stop = uicontrol(                                    ...
   'Parent',h,                                    ...
   'Style', 'pushbutton',                         ...
   'Units', 'normalized',                         ...
   'String', 'Stop',                              ...
   'FontSize',12,                                 ...
   'FontWeight','bold',                           ...
   'Position', [0.93 0.089 0.055 0.042],                   ...
   'Callback', 'set(findobj(''Tag'',''Play''),''Userdata'',0)');

%====================================
% Initialize CLOSE button
%====================================

h_close = uicontrol(                                   ...
   'Parent',h,                                    ...
   'Style', 'pushbutton',                         ...
   'Units', 'normalized',                         ...
   'String', 'Close',                             ...
   'FontSize',12,                                 ...
   'FontWeight','bold',                           ...
   'Tag','Close',                                 ...
   'Position', [0.93 0.0285 0.055 0.042],                  ...
   'Callback', 'beamctf(''CloseBEAM'')');

PARA(4)=findobj('Tag','Play');
PARA(5)=findobj('Tag','Close');
PARA(6)=findobj('Tag','saturationlabel');

PARA(10)=findobj('Tag','markx');
PARA(11)=findobj('Tag','marky');
PARA(12)=findobj('Tag','movex');
PARA(13)=findobj('Tag','movey');
PARA(14)=findobj('Tag','fwhhx');
PARA(15)=findobj('Tag','fwhhy');
PARA(16)=findobj('Tag','waistx');
PARA(17)=findobj('Tag','waisty');

PARA(20)=findobj('Tag','fmarkx');
PARA(21)=findobj('Tag','fmarky');
PARA(22)=findobj('Tag','fmovex');
PARA(23)=findobj('Tag','fmovey');
PARA(24)=findobj('Tag','ffwhhx');
PARA(25)=findobj('Tag','ffwhhy');
PARA(26)=findobj('Tag','fwaistx');
PARA(27)=findobj('Tag','fwaisty');

PARA(18)=findobj('Tag','lowlabel');
PARA(19)=findobj('Tag','highlabel');
PARA(28)=findobj('Tag','avglabel');
PARA(29)=findobj('Tag','rmslabel');

PARA(33)=findobj('Tag','textbarl');
```

Setwind.m

```
function setwind()

% Initialize Camera Window

global NUMCAM WINDOW ROW COLUMN DATA PROG
PREVIOUS REF PARA

if (PROG==1) | (PROG==2),

        windinit=init(NUMCAM,WINDOW(1),WINDOW(2),W
INDOW(1)+width(WINDOW(3)),...
                WINDOW(2)+height(WINDOW(3)));
        COLUMN=floor(width(WINDOW(3))+1);
        ROW=floor((height(WINDOW(3))+1)/2);
elseif PROG==3;

        [X,map]=imread(['j:\data\cern\image\',char(DATA(1)),'.t
if'],1);
        [ROW,COLUMN]=size(X);
else
        COLUMN=floor(width(WINDOW(3))+1);
        ROW=floor((height(WINDOW(3))+1)/2);
end;


set(PARA(7),'Enable','off');
set(PARA(4),'Enable','on');
set(PARA(2),'Enable','on');


graduate;
PREVIOUS=[COLUMN/2 ROW/2];
REF=PREVIOUS;
newview;
newcont;
```

Simbeam.m

```
function ans=simbeam(var)

% Simulate gaussian beam
% 1 -> TEM 00
% 3 -> TEM 10
% 5 -> TEM 01
% 7 -> TEM 11
% 2,4,6,8 -> add gaussian noise

global ROW COLUMN

X0=(0.2*rand+0.4)*ROW;
Y0=(0.2*rand+0.4)*COLUMN;
X=[1:ROW]';
```

```
Y=[1:COLUMN];
X=X-X0;
Y=Y-Y0;
W0X=COLUMN/(5+5*rand);
W0Y=ROW/(5+5*rand);
I=(0.8*rand+0.2)*(exp(-((X*ones(1,COLUMN)).*(X*ones(1,COLUMN))...

/(W0Y*W0Y)+(ones(ROW,1)*Y).*(ones(ROW,1)*Y)/(W0X*W0X)))).^2;
if (var>=5),
    I=I.*(((2*sqrt(2)*X*ones(1,COLUMN))/W0Y).^2);
end;
if (var==3) | (var==4) | (var>=7)
    I=I.*((2*sqrt(2)*(ones(ROW,1)*Y)/W0X).^2);
end;
MaxI=max(max(I));
if MaxI>1,
                    I=I/MaxI;
end;
if (mod(var,2)==0),
   I=imnoise(I,'gaussian', 0.003*rand*MaxI,0.003*rand*MaxI);
end;
ans=I;
```

Width.m

```
function ans=width(window)

  % Calculate the width of the window

  global PXLX PXLY

  ans=floor(sqrt(PXLY/PXLX)*window)-1;
```