User-Defined Method: CS-FLASH

Paravision C Code for Compressive Sensing

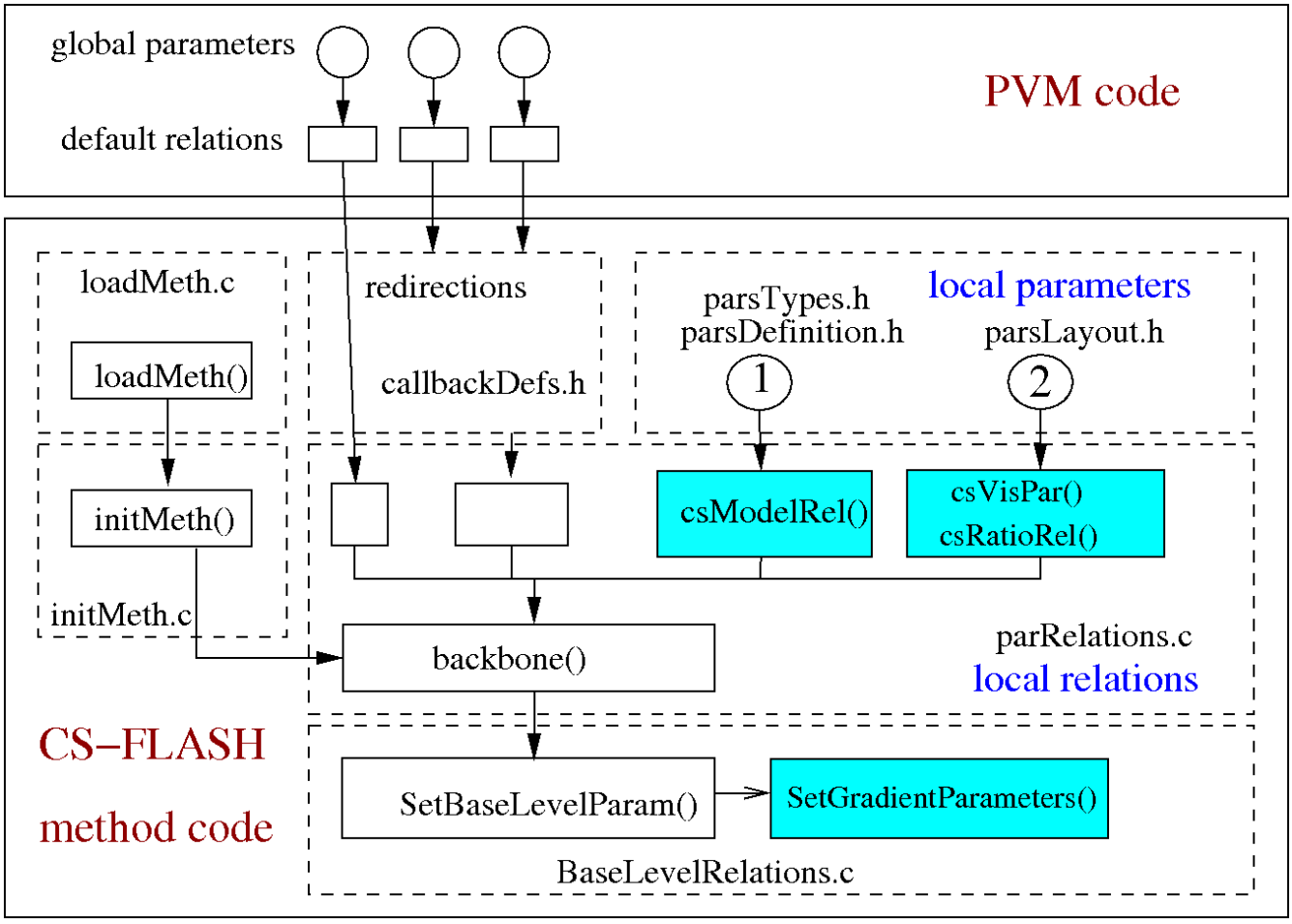
Programmed by Ming Yang, Dept. of Radiology, Univ. of Missouri – Columbia

Documented by Elliot Tan, Princeton University.

Modified by Y. Rosa Zheng, Dept. of ECE, Missouri University of Science and Technology

Reference: ParaVision Manual part D, Chapter 8 "Method Programming".

The CS-FLASH user-defined method is programmed based on the existing FLASH method distributed by BRUKER BioSpin. It is recommended that the user-defined method utilizes the internal PVM code to program the user method code with a structure shown in Fig. 8.2 of ParaVision Manual part D, Sec 8.3.15. The CS-FLASH method is copied from the FLASH method by the *copyMethod* script. (Ming: Is this true?) Several files are then reprogrammed according to Fig. 1, where major changes are made in the functions in the shaded boxes.



First, the variables needed for compressive sensing mode are defined in the three header files and their

relations to new functions are defined in backbone() of parRelations.c.

The summary of changes is listed in Table 1.

Table 1. Summary of changes of source files

Makefile of the FLASH method	Functionality	Makefile of the CS-FLASH method
flash.c	A glue file, always bearing the method's name	Modified to cs_flash.c
method.h	Link to necessary header files. Identical for all methods,	Copied. No modification.
relProtos.h	Link to function prototypes. Identical for all methods.	Copied. No modification.
parsTypes.h	User defined parameter types (e.g. enumerations)	Added an enum type *Compressive_Mode*
parsDefintion.h	Definitions of parameters.	Added two parameters: a *Compressive_Mode* type (CompressiveMode) and a double (CompRatio) CompressiveMode relates to CompressiveModeRel() CompRatio relates to csRatioRel()
parsLayout.h	Definition of "parclasses", including the Method-Class, which defines the layout in method editor GUI.	Added a parclass *CompressiveOptions* and inserted it into *MethodClass* *CompressiveOptions* contains 2 attributes: *CompressiveMode* and *CompRatio*
callbackDefs.h	Redirection of global (predefined) parameter relations.	Copied. No modification.
initMeth.c	code of the initMeth() function (required)	Copied. No modification.
loadMeth.c	code of the loadMeth() function (required)	Copied. No modification.
parsRelations.c	Defines parameter relations and other functions. Contains the backbone() function.	Added a local function CompressiveModeRel(), added local functions *CompRatioRel()* and *CompVisPar()*
BaseLevelRelations.c	functions for setting ACQP parameters for data acquisition, contains eight major functions: *1.SetBaseLevelParam ()* 2.*SetBasicParameters()* 3.*SetFrequencyParameters()* 4.*SetGradientParameters()* contains ATB_SetAcqTrims() 5. *SetInfoParameters()* 6. *SetMachineParameters()* 7. *SetPpgParameters()* 8. *SetACQ_obj_orderForMovie()*	Modify *SetGradientParameters()* to include two options: full and CS. In the CS option, add undersampling masks and modify ATB_SetAcqTrims() parameters to account for the gradient scaling
RecoRelations.c	functions for setting RECO parameters image reconstruction	Copied. No modification. Image reconstruction of CS is done in Matlab rather than in ParaVision

Successful compilation of the source code generates output files parsRelations.o, BaseLevelRelations.o, RecoRelations.o, cs_flash.o, cs_flash.so, cs_flash.4ch, and cs_flash.ppg.

Appenix: Five files have been changed from the original FLASH method:

- pars.Types.h
- parsDefinition.h
- parsLayout.h
- parsRelations.c
- BaselevelRelations.c

parsTypes.h

File Functionality

- Added new enum type of *Compressive_Mode* containing a "Gaussian" option and a "full" option
- A new parameter of *Compressive_Mode* type is defined in parsDefinition.h as *CompressiveMode*

Added/Edited Code

```
typedef enum
  {
        Gaussian,
        Full
  } Compressive_Mode;
```

parsDefinition.h

File Functionality

- Defines a Compressive Mode parameter and establishes the relation with the local function CompressiveModeRel()
- Defines a CompRatio parameter and relates to local function CompRatioRel()
- The two local functions are defined in parsRelations.c

Added/Edited Code

```
/* new parameters for Compressive Encoding */
```

```
Compressive_Mode parameter
{
        display_name "Compressive Mode";
        relations CompressiveModeRel;
}CompressiveMode;

double parameter
{
 display_name "Compressive Ratio";
 relations CompRatioRel;
 format "%.2f";
 units "/256";   this number shall be replaced by a variable
}CompRatio;
```

Additional Notes:

- display_name: determines what is displayed in Method Editor of Paravision software
- relations: determines the related function name
- format determines the number format of the input ratio
- units determines that the units of input ratio is the desired number of k-space lines sampled
- the name at the end of each definition, e.g. CompRatio, is the name of the parameter

parsLayout.h

File Functionality

- Defines a “parclass” called CompressiveOptions and includes it in MethodClass, which defines the layout in method editor
- The option is shown as “Ming's Compressive Options” seen in the method Editor of Paravision Software. Two input boxes “CompressiveMode” and “CompRatio” are also available in Editor.

Location of Edited Code (Two locations)

- New parclass function is added in succession to preceding parclass function
- “Compressive Options;” is added to the MethodClass parclass

Added/Edited Code

```
parclass
{
  CompressiveMode;
  CompRatio;
}attributes
{
  display_name "Ming's Compressive Options";
}CompressiveOptions;


parclass
{
  Method;
  PVM_EchoTime;
  PVM_RepetitionTime;
  PVM_NAverages;
  PVM_NRepetitions;
  PVM_ScanTimeStr;
  PVM_ExcPulseAngle;
  PVM_DeriveGains;
  RF_Pulses;
  Nuclei;
  Encoding;
  Sequence_Details;
  StandardInplaneGeometry;
  StandardSliceGeometry;
  Preparation;
  ScanEditorInterface;
  PPGparameters;
  ReconstructionOptions;
  CompressiveOptions; /* added for CS-FLASH*/
} MethodClass;
```

Additional Notes:

- When a parclass is defined, its name must be included in the MethodClass
- A parclass has to be defined before it is declared a member of another parclass

parsRelations.c

File Functionality

- void CompressiveModeRel(void) includes RecoCompVisPar() and the SetNewEncParam() from BaseLevelRelations.c which defines encoding parameters to control the MRI.
- void RecoCompVisPar(void) is used to determine when CompRatio is shown in the Method Editor, e.g. when sampling pattern is Gaussian, CompRatio is shown in Editor.
- void CompRatioRel(void) is used to set the default of CompRatio to 35 (hardcoded), and to restrict the CompRatio input to between 1 and 256.

Edited Code (Three locations)

- First block of modified code is used to turn on debugging, where 1 indicates "on" and 0 indicates "off"

```
#define DEBUG           1  //determines if messages should be activated
#define DB_MODULE       0 // determines if names of files of source should be printed
#define DB_LINE_NR      0 // determines if names of files of source should be printed
```

- Second block of modified code is to include RecoCompVisPar() and SetNewEncParam() in the function backbone()

```
/* at the end of the backbone method before DB_MSG((“<--backbone\n”));

 /* visibility of method specific compressive parameters */
  RecoCompVisPar();
  /* set compressive encoding */
  SetNewEncParam();
```

- Third block of modified code is to add relations for compressive sensing parameters in the Local Functions section

```
/* relations for compressive sensing parameters  */

void CompressiveModeRel(void)
{

  DB_MSG(("-->CompressiveModeRel"));

  RecoCompVisPar();
```

```
  SetNewEncParam();

  DB_MSG(("<--CompressiveModeRel"));
}

void RecoCompVisPar(void)
{
   DB_MSG(("-->RecoCompVisPar"));

   if(CompressiveMode==Full)
    ParxRelsHideInEditor("CompRatio");
   else
    ParxRelsShowInEditor("CompRatio");

   DB_MSG(("<--RecoCompVisPar"));
}


void CompRatioRel(void)
{
  DB_MSG(("-->CompRatioRel"));

  if(ParxRelsParHasValue("CompRatio") == No)
    CompRatio=35;

  CompRatio = MIN_OF(MAX_OF(1,CompRatio),256);

  DB_MSG(("<--CompRatioRel"));
}
```

BaseLevelRelations.c

File Functionality

- * Functions for setting ACQP parameters
- SetNewEncParamteres() is defined here

Variables Defined:

- PVM_Matrix: matrix size of viewing
- ACQ_size[1]: specifies matrix size in each direction
- ACQ_phase_encoding_mode: specifies phase ordering scheme used in acquiring raw data. In this case, use User_Defined_Encoding
- ACQ_phase_enc_start: specifies first phase encoding step acquired, it is set in the code, however its functionality is not used
- ACQ_spatial_size_1: specifies length of array in phase direction
- memcpy: C++ method, defined as - memcpy(void * destination, const void * source, size_t num)
- PVM_Encoding Values1 = the table of phase encoding steps scaled to the +/- 1 range → in this case it is copied to ACQ_spatial_phase_1
- ATB_SetAcqGradMatrix is used to set the gradient rotation based on PVM parameters of slice-geometry group

Location of Edited Code (Two locations)

- "SetNewEncParam();" is included in void SetBaseLevelParam(void) function, right before comment section beginning with "Sets parameters needed for multi-receiver acq. Overrides..."

```
void SetBaseLevelParam(void){

        …..

/* ---------------------------------------------------------------------
   Overide encoding parameters if compressive mode is on
   --------------------------------------------------------------------- */

  SetNewEncParam();
…..

}
```

- void SetNewEncParam() function is inserted before "Image sorting for movie mode" section, following the preceding functions such as void SetPpgParameters(void)

```
/*--------------------------------------------------*/
/*          Compressive Encoding                    */
/*--------------------------------------------------*/


void SetNewEncParam()
{
  DB_MSG(("-->SetNewEncParam\n"));

  if(CompressiveMode==Gaussian)
  {
  double GaussEnc[35] = {-0.515625, -0.40625, -0.3125, -0.296875, -0.265625, -0.234375, -0.1875, -
0.171875, -0.15625, -0.140625, -0.125,-0.109375, -0.078125, -0.0625, -0.046875, -0.03125,-0.015625,
0, 0.015625, 0.03125, 0.046875, 0.0625, 0.078125, 0.109375, 0.125, 0.140625, 0.15625, 0.171875,
0.1875, 0.234375, 0.265625, 0.296875, 0.3125,0.40625 ,0.515625};
    PVM_Matrix[1] = 35;
    ACQ_size[1] =35;
    ACQ_phase_encoding_mode[1] = User_Defined_Encoding;;
    ACQ_phase_enc_start[1] = -1.0; /* set, but no used */
    ACQ_spatial_size_1 = 35;
    memcpy(PVM_EncValues1, GaussEnc, sizeof(GaussEnc));
    ParxRelsCopyPar("ACQ_spatial_phase_1","PVM_EncValues1");
  }


  ATB_SetAcqGradMatrix( PVM_NSPacks, PVM_SPackArrNSlices,
                        PtrType3x3 PVM_SPackArrGradOrient[0],
                        PVM_ObjOrderList );


  if( PVM_ErrorDetected == Yes )
  {
    UT_ReportError("SetGradientParameters: In function call!");
    return;
  }

  ACQ_scaling_read  = 1.0;
  ACQ_scaling_phase = 1.0;
  ACQ_scaling_slice = 1.0;

  ACQ_rare_factor = 1;
```

```
 ACQ_grad_str_X = 0.0;
 ACQ_grad_str_Y = 0.0;
 ACQ_grad_str_Z = 0.0;


 strcpy(GRDPROG, "");

 ATB_SetAcqTrims( 10,
            PVM_ExSliceGradient,          /* t0 */
            (-PVM_ExSliceRephaseGradient),/* t1 */
            (-PVM_ReadDephaseGradient ),   /* t2 */  check here
            PVM_2dPhaseGradient,          /* t3 */
            (-PVM_3dPhaseGradient),       /* t4 */
            PVM_ReadGradient,            /* t5 */
            ReadSpoilerStrength,         /* t6 */
            (-PVM_2dPhaseGradient),       /* t7 */ check here
            PVM_3dPhaseGradient,          /* t8 */
            SliceSpoilerStrength         /* t9 */
          );

 if( PVM_ErrorDetected == Yes )
 {
  UT_ReportError("SetGradientParameters: In function call!");
  return;
 }


DB_MSG(("<--SetNewEncParam\n"));
}
```

Notes:

- First “if statement” is the actual Compressive Encoding code; everything after is turning on the gradients
- Currently hardcoded, code likely will be modified in the future to do away with hardcoding

Results:

- Copies hardcoded input of 35 phase encoding steps and sets ACQP parameters according to hardcoded input.

- GaussEnc array holds the values where ky lines are sampled if Gaussian option is selected in “Ming's Compressive Sensing Options”, code allows the array to be read by Paravision software.