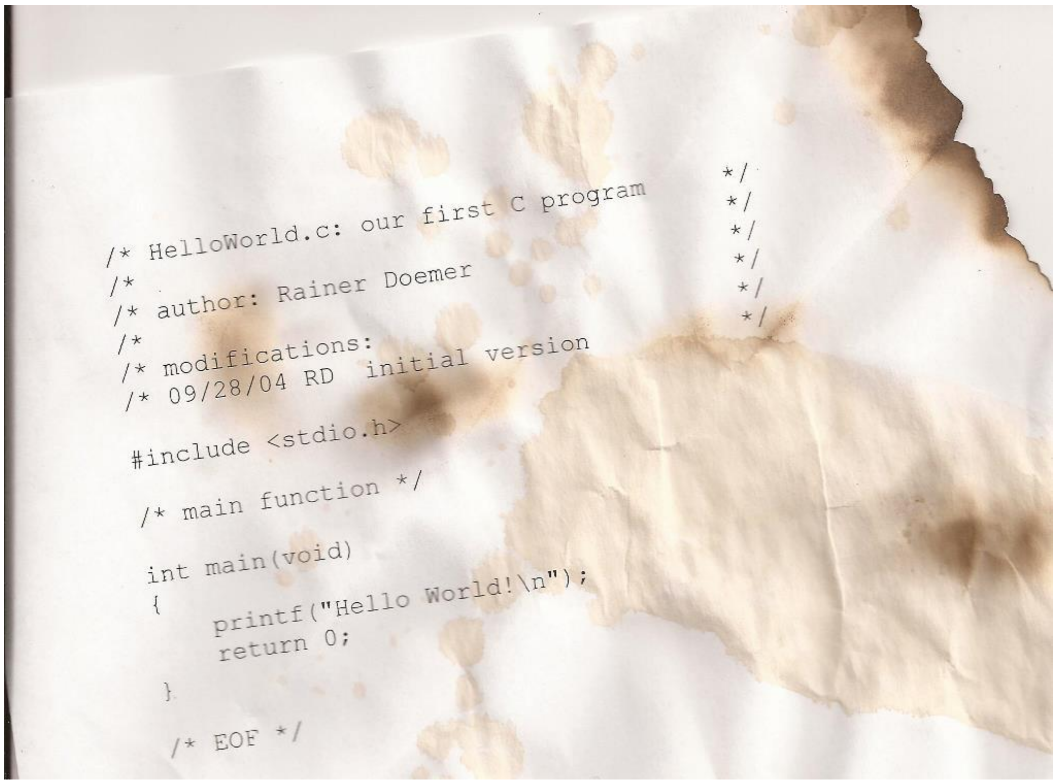


```
/* HelloWorld.c: our first C program       */
/*                                          */
/* author: Rainer Doemer                    */
/*                                          */
/* modifications:                           */
/* 09/28/04 RD  initial version             */

#include <stdio.h>

/* main function */

int main(void)
{
    printf("Hello World!\n");
    return 0;
}

/* EOF */
```

Optical Character Recognition

Software Specification

Version 1.1

Team 3

Quan Chau

Hanchel Cheng

Kevin Duong

Jamie Lee

Ryan Morrison

Eric Rodriguez

Andrew Trinh

Table of Contents

Glossary

Digital image processing (DIP) – functions used to edit the image such as sharpening, rotation, resizing, and etc.

Optimal Character Recognition (OCR) – program that matches characters on an input image to produce a text result

Post-processing – editing of text file produced by the OCR

Pre-processing – editing done to the input image to prepare for OCR program

1: Software Architecture Overview

1.1 Main data types and structures

- Image
- ImageList and ImageEntry
- ObjectHandle
- ViewHandle
- CharProbability
- CharProfile

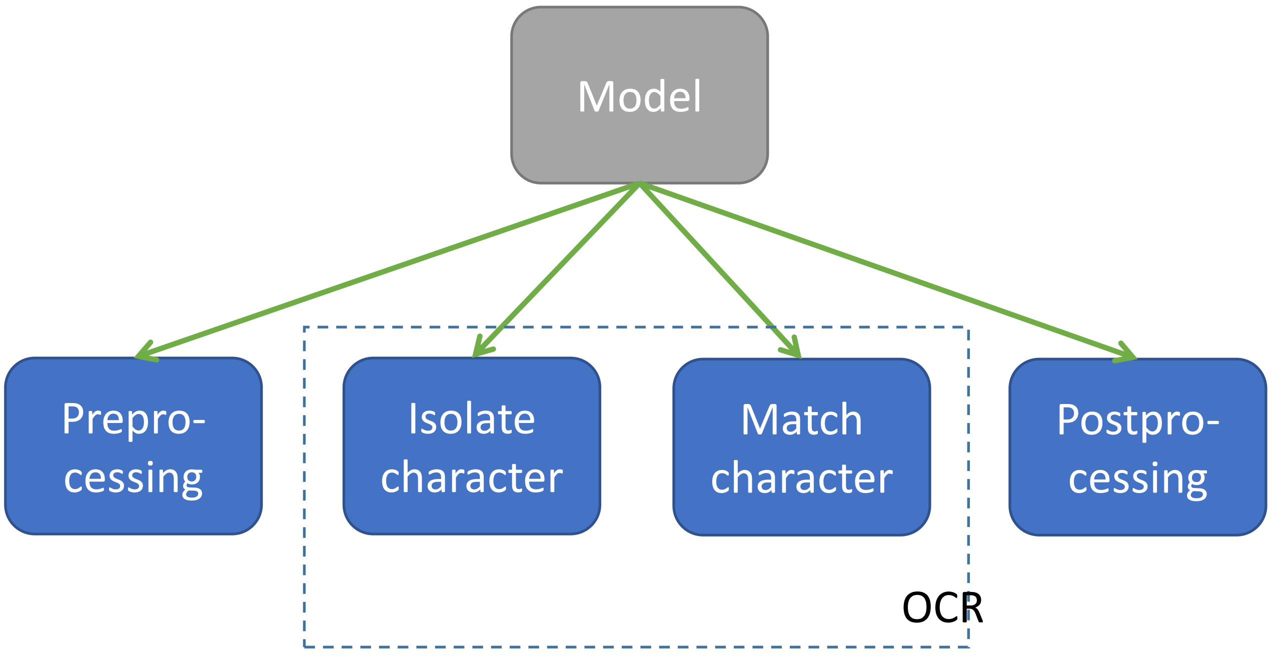
1.2 Major software components

The program will be splitted into three major components: View, Control, and Model.

1.2.1 Model:

Model is responsible for all logic operations and image processing. The model module is divided into four submodules:

- Preprocessing
- Isolate character
- Match character
- Post processing



1.2.2 Control:

Control module processes events, updates the states and call appropriate functions.

1.2.3 View:

View module displays the image and text to the users and receives events initiated from keyboard or mouse by user.

1.3 Module interfaces

1.2.1 Model:

IMAGE * RotateImage (IMAGE * image, int degree);

IMAGE * BlackWhiteImage (IMAGE * image);

IMAGE * ColorFilter (IMAGE * image, int area_x1, int area_y1, int area_x2, int area_y2, int NewR, int NewG, int NewB);

UT_string * PerformOCR (IMAGE* image);

1.2.2 Control:

Control_Initialize()

Control_MainLoop()

Control_ProcessEvents()

Control_CleanUp()

1.2.3 View:

DrawMainWindow()

DrawRotatePanel()

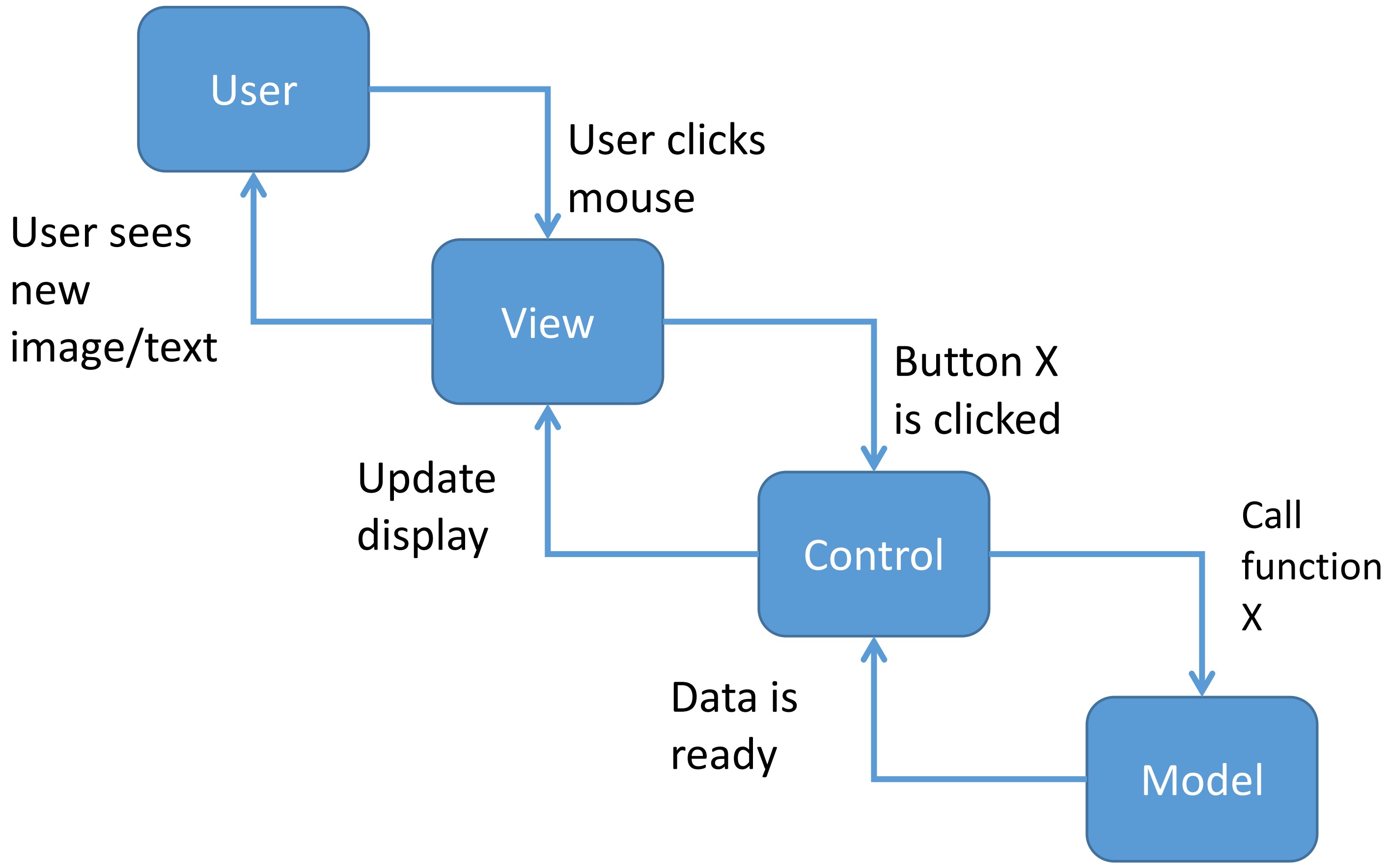
DrawWrinkleRemovalPanel()

DrawStainRemovalPanel()

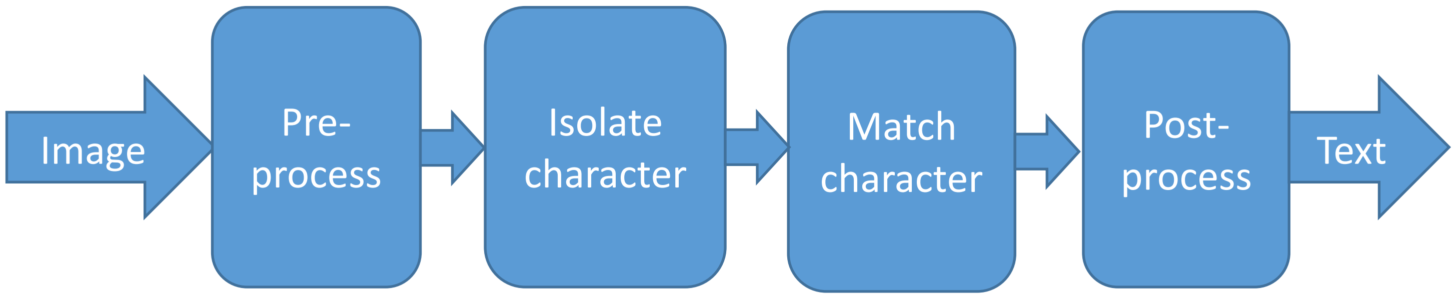
DrawCropImage()

1.4 Overall program control flow

General view:



Flow chart from an input image to final text result:



2: Installation

2.1 System requirements

- Hardware: PC Hardware (x86_64 server)
- Operating system: Linux OS (RHEL-6-x86_64)
- Dependent third party software:
  1. gcc
  2. GNU make
- Dependent libraries:
  1. GTK+-2.0 or GTK+-3.0 for graphical user interface
  2. Netpbm library for image processing
  3. Uthash library for dynamic hash table, array and string

2.2 Setup and configuration

- GTK+ library installation: details are found at www.gtk.org
- Netpbm library installation: details are found at netpbm.sourceforge.net
- Uthash library installation: not necessary because it's already included in the package
- The software comes in a tar.gz package. After downloading, extract the package by running:
  ```
  tar –zxvf OCR.tar.gz
  ```
- Change into the directory by running:
  ```
  cd OCR
  ```
- Compile the program by running:
  ```
  setenv PKG_CONFIG_PATH /usr/share/pkgconfig
  make OCR
  ```

2.3 Uninstalling

- Change into the directory by running:
  ```
  cd OCR
  ```
- Compile the program by running:
  ```
  make clean
  ```

3: Documentation of OCR Modules and Interfaces

3.1 Detailed description of data structures

- Image
  This data structure hold Red, Green and Blue intensity of a pixel in an image.
  ```
  typedef struct {

          unsigned int Width, Height;

          unsigned char *R, *G, *B;

      }IMAGE;
  ```
  This structure has three states:
  - Color: R, G and B have distinct values from 0 to 255
  - Grayscale: R, G and B have same value from 0 to 255
  - Black and white: R, G and B all are 0 (black) or 255 (white)

- ImageList and ImageEntry
  These two structures are used to have a dynamic array of Image struct. They are imported from last quarter assignments.
  ```
  struct ImageList

      {  int  Length;

        IENTRY *First, *Last;

      } ;

      struct ImageEntry

      {  IENTRY *Next, *Prev;

        ILIST  *List;

        IMAGE  *Image;

      };
  ```

- ObjectHandle:
  This structure holds the name of an object and the GtkWidget * of that object to allow communication between View and Control.
  ```
  typedef struct {

          char Name[MAX_HASH_KEY_LENGTH];

          GtkWidget * Widget;

          UT_hash_handle HashByName;          /*to allow search an object by name*/

          UT_hash_handle HashByWidget;        /*to allow search an object by pointer value*/
  ```

} ObjectHandle;

- ViewHandle:
  This structure holds the dictionary (hashtable) of many ObjectHandle. ObjectHandle can be queried by either Name or GtkWidget * value.
  typedef struct {

  ObjectHandle * ObjectListByName, * ObjectListByWidget;

  } ViewHandle;

- CharProbability

This structure holds an ascii value and the probability that this ascii value match this single character image
typedef struct {

char Char;

int Probability;

} CharProbability;

- CharProfile
  This structure represents a character spot in the final text string. It contains an array of CharProbability and CharChosen

  typedef struct {

  UT_array * CharChoices;

  char CharChosen;

  } CharProfile;

3.2 Detailed description of pre- and post-processing, and OCR functions and parameters

Preprocessing
IMAGE * CropImage(IMAGE *, int x1, int y1, int x2, int y2)

Description: take an image and crop it from coordinate (x1, y1) to (x2, y2). First the function will rearrange two coordiantes to get the upper left and lower right corner of the area to crop. Afterwards, it will allocate memory space for new image with new size then set pixel of new image to match those pixels of the cropped area on the original image.

IMAGE * Rotate(IMAGE *, int ClockwiseDegree)

Description: This function will rotate the image clockwise direction by a given degree.

IMAGE * Resize(IMAGE *, int new_x, int new_y)

Description: This function will resize the image into new size. The algorithm we use is bilinear interpolation. The detailed description can be found here:
http://en.wikipedia.org/wiki/Bilinear_interpolation

IMAGE * ColorFilter(IMAGE *, int x, int y, int area_x1, int area_y1, int area_x2, int area_y2, int NewPixelValue)

Description: This function is intended to solve coffee stain. It will go through the area from (x1, y1) to (x2, y2) and find all the pixel that has color similar to pixel at (x,y) and change its R,G,B intensities into NewR, NewG and NewB

OCR:
ImageList * IsolateCharacter(IMAGE *, FontEnum, FontSizeEnum, ScanResolutionEnum)

Description: This function takes in one IMAGE * and breaks it into multiple IMAGE *, each contains a character. The font type, font size and scan resolution determine how the image is split.

UT_array * MatchCharacter(ImageList *, FontEnum, FontSizeEnum, ScanResolutionEnum)

Description: This function will return an array of CharProfile. Each CharProfile represents a character in the final string and contains all possible characters that this single image matches. The font type, font size and scan resolution determine which font template to select and how the font template is stretched before comparison algorithm works.

Post processing:
UT_string * Postprocess(UT_array * CharProfileArray)

Description: This function takes input an array of CharProfile and output a string. The function has a dictionary of C syntax and tries to match the string with the keywords of C language

3.3 Detailed description of input images and output text

Input images:

This software supports the following input images format:

- PNG
- BMP
- PPM
- JPEG/JPG

The conversion from formats other than PPM into PPM uses netpbm library. In particular, the APIs we use from netpbm are:

- pngtoppm
- bmptoppm
- jpegtoppm

After the image is converted into PPM, the file will be read into IMAGE struct defined above and kept in the program memory to be processed

Output text:

Output text will be an array of type char ending with NULL. The char array will contain new line characters as well as normal characters. Here is an example:

H	e	l	l	o	(space)	W	o	r	l	d	\n	E	n	d	\0

4: Testing Plans

Overview:

Module	**Function**	**Input**	**Output**	**Description**	**Phase**
Preprocess	Crop	IMAGE	IMAGE	Fully functional	Alpha
	Rotate	IMAGE	IMAGE	Fully functional	Alpha
	Resize	IMAGE	IMAGE	Fully functional	Alpha
	ColorFilter	IMAGE	IMAGE	Fully functional	Alpha
OCR	IsolateCharacter	IMAGE	ILIST	CourierNew, 12pt, 300DPI	Alpha
		IMAGE	ILIST	LucidaConsole, 10pt, 300DPI	Beta
	MatchCharacter	ILIST	UT_array * CharProfileArray	CourierNew, 12pt, 300DPI	Alpha
		ILIST	UT_array * CharProfileArray	LucidaConsole, 10pt, 300DPI	Beta
Postpro-cessing	PostProcess	UT_array * CharProfileArray	UT_string *	Take only the character with highest probability	Alpha
		UT_array * CharProfileArray	UT_string *	Compare with dictionary	Beta
GUI	Layout + Event	Keyboard + mouse	Event	All operations	Alpha
				Select coordinate by enter number	Alpha
		Keyboard + mouse	Event	Select coordinate by mouse	Beta

Details:

Preprocessing:

- Crop: Take a picture and crop it, must make sure that it takes coordinates from any order
- Rotate: Take a picture and rotate it with any integer degree from 0 to 360. Will test it by issuing random integer degree many times
- Resize: Take a picture and resize it to any new scale. Will test it by issuing random large integer as new scale
- ColorFilter: Take a picture and filter only a certain parts to either black or white using a pre-defined threshold. Will test it by issuing random coordinates

OCR:

- Isolate character: Will test it by reading different images and show the lines to cut into small images. This way is faster since we don't have to save a lot of images and can see the big picture of where the lines are cut

- Identify character: Will take random input from the Isolate Character and try to identify it.

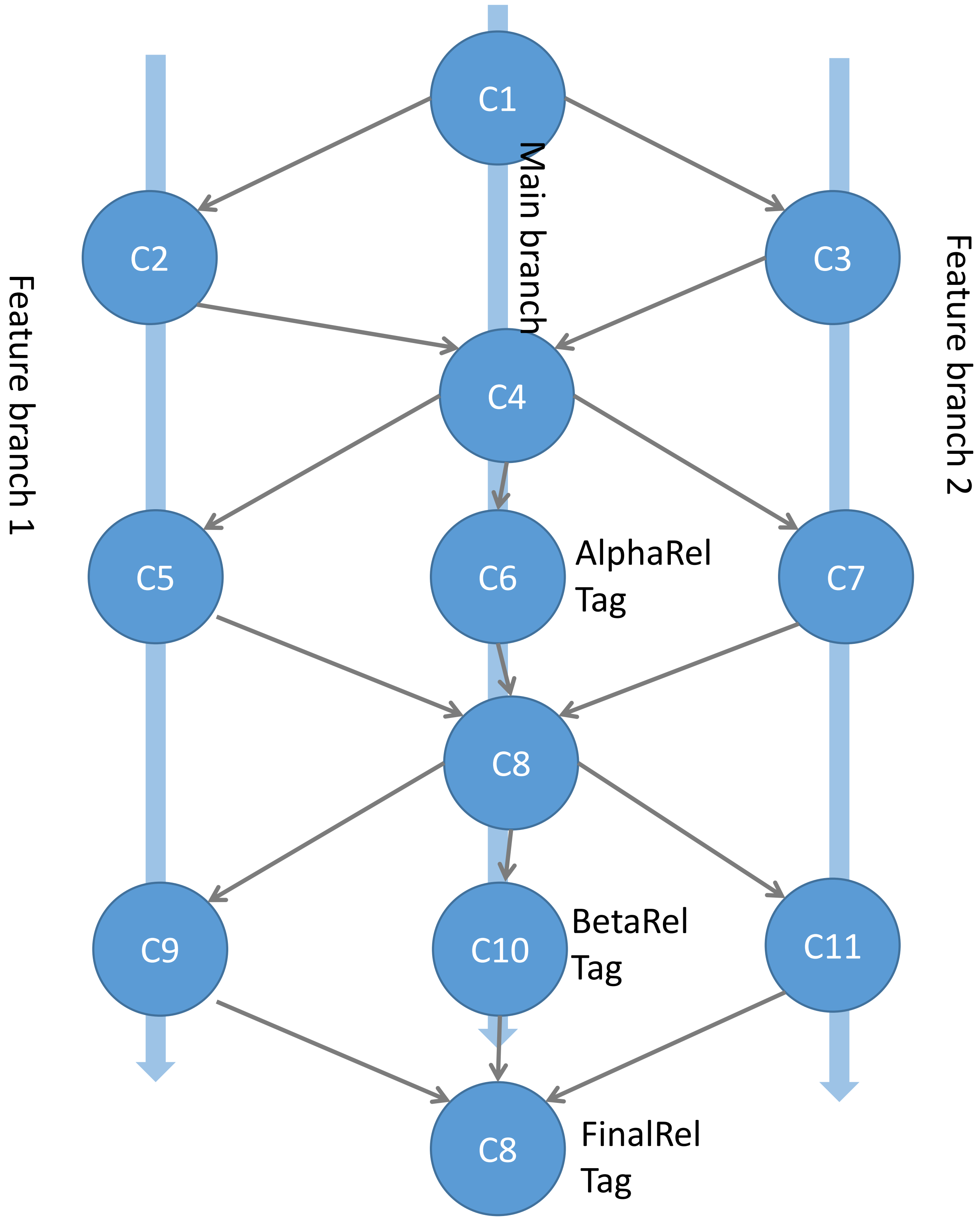
Postprocessing:

- PostProcessing: Generate a random sequence of character probability and takes only character with largest probability. In next phase will try to match the same sequence with as many C keywords as possible.

5: Development Plan and Timeline

	GUI (Ryan + Quan)	Control (Quan)	Preprocessing (Eric)	Isolate character (Hanchel + Jamie)	Identify character (Andrew)	Post processing (Kevin)	Image input (Jamie)
Week1 (by Feb 17)	Pseudo code + try out a real software	Pseudo code + try out a real software	Pseudo code + try out a real software	Pseudo code + try out a real software	Pseudo code + try out a real software	Pseudo code + try out a real software	Pseudo code + try out a real software
Week2 (by Feb 24)	Layout ready	Main flow ready	DIPs + Rotate by any degree	CourierNew, 12pt, 300DPI only	CourierNew, 12pt, 300DPI only	Choose character with highest match	Any file type input
Week3 (by March 3)	Interaction ready	combine many pages, allow external text editor	Solve wrinkle + coffee spot	LucidaConsole, 10pt, 300DPI only	LucidaConsole, 10pt, 300DPI only	Match word with C syntax dictionary	Update if necessary
Week4 (by March 10)	Update if necessary	Update if necessary	Update if necessary	New settings if necessary	New settings if necessary	Run C compiler, make it as compilable as possible	Update if necessary
Week5 (by March 17)	Fix small bug from last release	Fix small bug from last release	Fix small bug from last release	Fix small bug from last release	Fix small bug from last release	Fix small bug from last release	Fix small bug from last release

6: Branching plan



Branching plan explanation:

1) C1, C2, C3… represents groups of commits on CVS
2) The black arrows coming out from one common commit represents creating a new branch
3) The black arrows coming into one common commit represents merging
4) Blue arrow represents a feature branch, such as PostProcessing branch and PreProcessing branch

Benefits of this branching plan:

1) Avoid pitfall of long-time unmerged branches:
   - The longer the branch stays unmerged, the more chance there are more conflicts to merge
   - This plan forces feature branches to merge frequently (every week) to stay up to date with changes in main branch
2) Allow team members to focus on developing features instead of merging
3) Merge stable codes only
4) Merging and developing are done simultaneously
5) Avoid rush to debug new feature to include in release

Back Matter

1. Copyright

2. Error message:
   a. Lines boundaries not yet selected: This error means the user didn't provide the line boundaries yet
   b. Invalid image format: This error means the image selected doesn't have a supported format
   c. Unable to perform OCR: This error means the preprocessed image isn't clear enough for the OCR to detect any characters. To solve this problem, user will have to do some more preprocessing on image before retrying OCR

Index