Projet informatique Mastère PPMD 2017-2018

« RTK BASE » Développement d'un module de calcul automatique des coordonnées de la base

Manuel programmeur

ENSG Géomatique

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

Commanditaires :

- **Jean-Yves PERRIN**
- **Francklin N'Guyen**

Auteur :

Saif AATI

Février 2018

Table des matières

Liste des Figures

Introduction

Ce sujet fait partie d'un projet plus large mis en place par J.-Y. Perrin et F. N'Guyen dont l'objectif d'obtenir un système de positionnement temps réel à moindre coût. Ce dernier est partagé avec la communauté open source sur Github [1]. Il est prévu pour les systèmes type UNIX et formaté pour convenir au mieux à un matériel type Raspberry Pi équipé d'un écran tactile.

Tous les résultats de calculs sont issus des fonctions de RTKLIB et donc leur précision y est directement liée.

En effet, ce logiciel est une alternative aux solutions constructeurs telles que Trimble et Leica qui proposent des produits complets et performants mais inaccessibles aux particuliers et petits professionnels [2].

L'objectif du projet est d'ajouter un module permettant le calcul automatique des coordonnées de la base.

Pour ce faire, l'opérateur installe le récepteur pour une acquisition statique. Dès que suffisamment d'observations ont été collectées, le récepteur se connecte à un réseau de stations permanentes (en France le RGP), télécharge les données des n stations les plus proches, calcule sa position et peut ainsi être configuré automatiquement en base RTK.

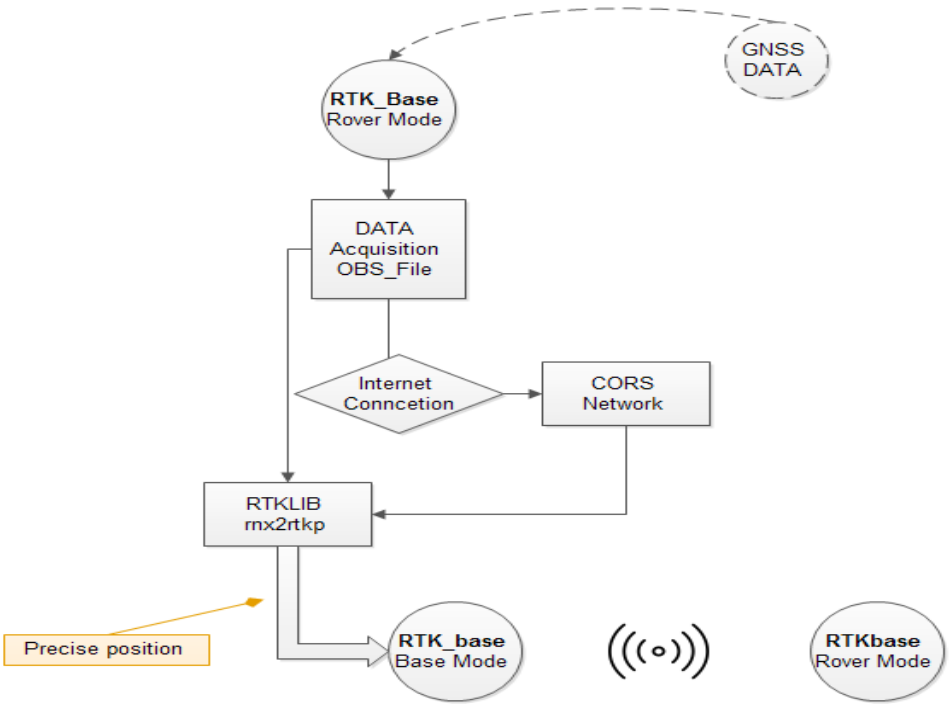

Figure 1 :Fonctionalités du programme

Ce programme permet d'assurer les fonctionnalités suivantes :

- Récupérer le fichier d'observation et une position approchée de la station ;
- Déterminer les stations les plus proches ;
- Télécharger les données nécessaires au post traitement ;
- Faire un calcul avec RTKlib en mode ligne de commande (rnx2rtkp) ;

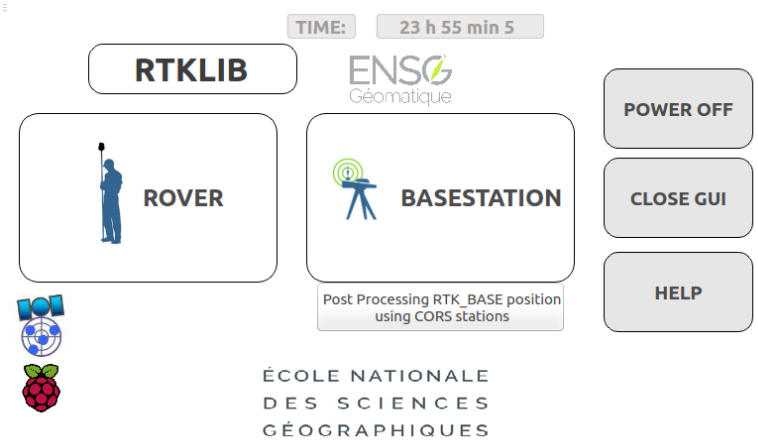

Figure 2 : Ecran d'accueil de RTKBASE

Les différents libraires et fichiers

-	approxcoord.h	approxcoord.cpp	
-	coord_coverter.h	coord_coverter.cpp	
-	downloader.h	downloader.cpp	
-	gpstime.h	gpstime.cpp	
-	mydialog.h	mydialog.cpp	mydialog.ui
-	mymessagehandler.h	mymessagehandler.cpp	
-	processing.h	processing.cpp	
-	rnx2rtkp.h	rnx2rtkp.cpp	
-	station.h	station.cpp	

Détail des libraires et classe

Point de départ, le clique sur le bouton « Post Processing RTK_Base Psitionusing CORS stations », fait appel à la classe : class MyDialog

class MyDialog

Méthodes de MyDialog :

- void on_pushButton_run_rnx2rtk_process_RGP_clicked() : la connexion au bouton permet :
  - D'appeler la fonction processing() de la librairie processing.h
  - D'appeler la fonction qInstallMessageHandler() de la librairie qInstallMessageHandler.h
  - D'appeler la fonction on_progressBar_valueChanged() :pour afficher l'avancement du calcul
  - Créer un fichier log.txt qui contient le rapport du calcul
  - Afficher le rapport final sur la fenêtre de post traitement
- void on_progressBar_valueChanged(int value): widget de QProgressBar permet de fournir une barre de progression horizontale.

Librairie_Processing

La fonction processing(), assure la chronologie du calcul :

1- Vérification des packages
2- Installation des packages
3- Crée des objets des classes Rnx2rtkp, Downloader, Approxcoord, Station, Gpstime
4- Conversion du fichier d'observation (.ubx) en format rinex (.obs) via la fonction conversion() de la classe Approxcoord
5- Gérer les époques de disponibilités des données sur le FTP du RGP
6- Vérifier la connexion internet et télécharger le fichier de coordonnées des stations RGP (coorrgf93_2018-020.txt) via les méthodes de la classe Downloader
7- Déterminer les stations les plus proches, via la fonction neareststation() de la classe Station
8- Télécharger les données nécessaires pour accomplir le calcul du post traitement, via la fonction do_downloader() de la classe Downloader
9- Générer le fichier de configuration (configuration.conf) via la fonction configuration_file() de la classe Rnx2rtkp
10- Effectuer le calcul rnx2rtkp via la fonction rnx2rtkp() de la classe Rnx2rtkp
11- Etablir le fichier des résultats de calcul (resultats.pos) via la fonction final_results() de la classe Rnx2rtkp

class Approxcoord

Méthodes de Approxcoord :

- void Approxcoord::conversion() : Permet de convertir un fichier d'observation brut type ublox en fichier de format rinex.
- void Approxcoord::approx_coord() : Permet de lire le fichier d'observation (rover.obs) et renvoie : la position approximative , début /fin de l'observation

class Coord_coverter

Méthodes de Coord_coverter:

- QVector<double> ecef_to_geo( ): Permet de convertir les coordonnées cartésiennes en coordonnées géographique
  - Entrée : Un tableau à trois éléments contenant x, y, z en mètres
  - Sortie : Array contient lat et lon en radians, et l'altitude en mètre
- QVector<double> geo_to_ecef( )

class Downloader

Méthodes de Downloader:

- void do_downloader() : Permet de vérifier la connexion internet et télécharger les fichiers
  - Entrée :
    - url : Lien de téléchargement
    - file_name: Nom du fichier à télécharger
    - saving_path: Chemin d'enregistrement (Répertoire work)
  - Sortie : Répondre au statut du téléchargement

- void unzip_file():Permet de décompresser les fichiers
  - Entrée :
    - file_name: Nom du fichier
    - saving_path: Chemin d'enregistrement (Répertoire work)
  - Sortie : Fichier non compressé

- void uncmpress_hatanaka(): Permet la décompression logique (Hatanaka) des fichiers

Attributs :

- QString url: Lien de téléchargement
- QString file_name : Nom du fichier à télécharger
- QString saving_path: Chemin d'enregistrement (Répertoire work)

Librairie_myMessageHandler

La fonction myMessageHandler () permet la gestion des messages : une fonction qui imprime des messages de débogage, des avertissements, des messages d'erreur critiques et fatales.

class Rnx2rtkp

Méthodes de Rnx2rtkp:

- void configuration_file() : Permet de générer le fichier de configuration pour le processus de calcul RNX2RTKP
- void rnx2rtkp() : Permet d'exécuter le calcul rnx2rtkp
  - Entrée :
    - Fichier d'observation ---> rover.obs
    - Fichier d'observation de la station ---> [sitejjjs.aad]
    - Fichier de navigation GPS de la station ---> [sitejjjs.aan]
    - Fichier de navigation GLONASS de la station ---> [sitejjjs.aag]
    - Fichier de configuration ---> Configuration.conf
  - Sorite
    - Fichier contenant les résultats des calculs ---> i-resultats.pos

- void final_results ():
    - Création du fichier OUTPUT final du processus rnx2rtkp ---> resultats.pos
    - Ce fichier contient les résultats des calculs RNX2RTKP avec les différentes stations téléchargées
    - Utiliser la méthode de la médiane pour calculer les coordonnées finales de RTK_BAse sation

Attributs :

- QString path: chemin du répertoire de travail ../work/
- QString station_obs_file: nom du fichier des observation de la station
- QString station_nav_file: nom du fichier de navigation (GPS) de la station
- QString station_g_file: nom du fichier de navigation (GLONASs) de la station
- QVector<QString> X_Y_Z_ecef_final: Coordonnées de la station RTK_Base

class Station

Méthodes de Station:

- void nearestsation() : Permet de lire le fichier des coordonnées des stations et trier les stations
    - Entrée :
        - Fichier des coordonnées des stations coorrgf93_2018-020.txt
        - static const float d_max
    - Sorite
        - QVector<QString> vect_name
        - QVector<double> vect_dist
        -
- QVector<QString> data_file_nearest_sation():Permet de gérer les époques de diffusion des données
    - Entrée :
        - Doy : jour de l'année
        - yyyy : l'année
        - TIME_OF_FIRST_OBS
        - TIME_OF_LAST_OBS
        - Le numéro de la station
    - Sortie :
        - QVecteur qui contient :
            - Nom de la station
            - Nom du fichier d'observation
            - Nom du fichier de navigation (GPS)
            - Nom du fichier de navigation (GLONASS)
            - Le lien de téléchargement

- QString Corrdstation_ftp(): Permet de générer le lien de téléchargement du fichier des coordonnées des stations
    - Entrée :
        - Doy : jour de l'année
        - yyyy : l'année
    - Sortie :
        - QString corrdstation_ftp
- void station_data(): Permet de lire le fichier d'observation de chaque station.
        - Entrée :
            - Fichier d'observation de la station [sitejjjs.aad]
            - Emplacement du fichier
        - Sortie :
            - QString _antenna_type_station
            - QVector<QString> _coord_antenna
            - QVector<QString> _coord_station
            - QVector<QString> vect_name
            - QVector<QString> vect_domes
            - QVector<double> vect_X,vect_Y,vect_Z

Attributs :

- d_max : le rayon de recherche max des stations à 100km
- nb_stat_min : Le nombre minimal des stations à télécharger pour lancer un calcul (fixé à 4 stations)
- QString corrdstation_ftp : Lien de téléchargement du fichier des coordonnées des stations
- QString _antenna_type_station: Type d'antenne de la station
- QVector<QString> _coord_antenna: Coordonnées de l'antenne
- QVector<QString> _coord_station: Coordonnées de la station
- QVector<QString> vect_name: Nom des stations les plus proches
- QVector<QString> vect_domes: Numéro international des stations les plus proches
- QVector<double> vect_X,vect_Y,vect_Z,vect_dist : Coordonnées et distances des stations par rapport à la postion de station RTK_Base.
- QVector<double>X0 : Coordonnées approchées de la station RTK_Base