Universidade de São Paulo
Instituto de Astronomia, Geofísica e Ciências Atmosféricas
Departamento de Astronomia

Rodrigo Trevisan Massera

Desenvolvimento e Aprimoramento do *Software* de Controle do instrumento BTFI (*Brazilian Tunable Filter Imager*)

São Paulo
2015

Rodrigo Trevisan Massera

Desenvolvimento e Aprimoramento do *Software* de Controle do instrumento BTFI (*Brazilian Tunable Filter Imager*)

Trabalho de Conclusão de Curso apresentado ao Departamento de Astronomia do Instituto de Astronomia, Geofísica e Ciências Atmosféricas da Universidade de São Paulo

Área de Concentração: Astronomia
Orientadora: Prof.ª Dr.ª Claudia Lucia Mendes de Oliveira

São Paulo

2015

Dedico este trabalho à minha família, a meus amigos e a todos que me deram suporte nesta jornada.

Agradecimentos

A minha família;

A minha orientadora $Prof^a$ Dr^a Claudia de Oliveira;

Aos pesquisadores e colegas Bruno Quint e Denis Andrade;

A todos os meus colegas do curso que me deram apoio;

À FAPESP, pelo apoio financeiro, sob o projeto n^o: 2012/17223-0;

Esta tese/dissertação foi escrita em LaTeX com a classe IAGTESE, para teses e dissertações do IAG.

Resumo

Este trabalho consiste do desenvolvimento e aprimoramento do software de controle projetado para o instrumento BTFI (*Brazilian Tunable Filter Imager*), que está instalado atualmente no telescópio SOAR, no Chile.

Ao longo dos quatro semestres, várias melhorias foram implementadas, como um cabeçalho mais completo para as imagens, uma melhoria na eficiência, que permite um melhor aproveitamento do tempo do telescópio produzindo uma maior quantidade de dados. Além do emprego do controle de versão, facilitando a organização do projeto. Todas estas alterações permitiram otimizar a utilização do instrumento.

Abstract

This work consisted on developing and improving the control software designed for the BTFI (Brazilian Tunable Filter Imager) instrument, which is currently installed on the SOAR telescope in Chile.

Over the past couple of years, several improvements were made, such as a more complete header for the images, an upgrade in efficiency, which allows for better use of observational time, producing larger amounts of data, employment of version control, facilitating the project organization. All those changes contributed to the optimization of the instrument.

Lista de Figuras

1.1 Foto do instrumento 18
1.2 Exemplo da Interface de blocos do LabVIEW 19
1.3 Esquema ilustrando um dos primeiros programas elaborados para leitura de dados do BTFI. 19
1.4 Diagrama exemplificando a comunicação entre o usuário e o instrumento . 20
1.5 Janela principal do software de controle do instrumento, denominado Kafka. 21

2.1 Exemplo de um cabeçalho contido na imagem .fits. 24
2.2 Figura comparando o tempo real que se gasta para fazer cada imagem versus o tempo ideal. 27
2.3 Ilustração das informações do TCS (à direita) e sobre as rodas de filtro e máscara (à esquerda). 29
2.4 Apresentação das configurações das máscaras e filtros estabelecidas por um determinado usuário. 29
2.5 O Log gerado pelo programa, da ordem dos eventos mais recentes para os mais antigos. 30
2.6 Fluxograma indicando o funcionamento do controle de versão empregando o conceito de "Mergers"e "Branches". 32

A.1 Cabeçalho atual completo gerado pelo software. 40

Lista de Tabelas

2.1 Tabela com os dados obtidos dos tempos de exposição em um scan 26
2.2 Tabela contendo o tempo perdido para cada configuração de scan. 27

Sumário

1. Introdução 17
1.1 BTFI (*Brazilian Tunable Filter Imager*) 17
1.2 O LabVIEW 17
1.3 O Software de Controle - Kafka 20

2. Desenvolvimento 23
2.1 O Cabeçalho 23
2.2 Melhorias na Eficiência 24
2.3 TCS, Roda de Filtros 28
2.4 Log do Sistema 30
2.5 O Emprego do Controle de Versão no Kafka 31

3. Conclusões 33

Referências 35

Apêndice 37

A. Cabeçalho Atual 39

Capítulo 1

Introdução

1.1 *BTFI (*Brazilian Tunable Filter Imager*)*

O chamado *"Brazilian Tunable Filter Imager"*, BTFI, (Taylor 2010) é um instrumento imageador óptico com filtro ajustável instalado no telescópio SOAR, no Chile, e será usado tanto para observações limitadas por *seeing* natural como para observações com óptica adaptativa. Atualmente o BTFI encontra-se em fase de testes e comissionamento no telescópio SOAR. O instrumento será dedicado a aquisição de cubos de dados espaciais e espectrais (com 2 dimensões espaciais e uma dimensão espectral) e abre novas e importantes possibilidades científicas para a comunidade brasileira de astronomia, desde o estudo de galáxias próximas e meio interestelar até investigações estatísticas de cosmologia. A concepção do instrumento utiliza duas novas tecnologias, já testadas com sucesso nos laboratórios, mas que ainda têm que ser testadas no céu, em um instrumento astronômico. O conceito do imageador e filtro ajustável Bragg (*imaging Bragg Tunable Filter* - iBTF), que constitui um dos braços do instrumento, utiliza um par de redes holográficas ou redes de difração de Bragg (*Volume Phase Holographic or Bragg Difraction Gratings* -VPHGs) enquanto que um novo conceito de *Fabry-Perot* (FP), instalado no segundo braço do instrumento, envolve o uso de tecnologia comercialmente disponível que permite um etalon (nome equivalente a *Fabry-Perot*) atuar em uma grande faixa de ordens de interferência. Estes detalhes podem ser verificados no site do projeto BTFI.

1.2 *O LabVIEW*

O LabVIEW da empresa NationalInstruments, é uma linguagem de programação voltada ao setor de automação industrial, com a facilidade de ter uma fácil interface e filosofia

Figura 1.1: Foto do instrumento

de "blocos" (que pode ser vista na Figura 1.2). Devido as vantagens proporcionadas, foi a escolhida para ser utilizada na confecção do software de controle do BTFI.

A versão deste programa é a de 2013 e qualquer pessoa que possua um vínculo com a USP do campus em São Paulo possui acesso a ele através do site LSI.

Para acessar, é necessário que o usuário esteja dentro do campus e possua um endereço de IP permitido pelo sistema. Lá encontram-se todos os arquivos necessários para a instalação e instruções de como utilizar as ferramentas.

A primeira etapa foi a de aprendizado da linguagem, sendo necessário ler os manuais e tutoriais disponíveis e aprender desde as tarefas mais básicas, até mais complexas. Contou-se com o suporte do Álvaro Calasans, engenheiro do projeto e principal desenvolvedor do software de controle do instrumento.

Um dos primeiros programas utilizados como treinamento, pode ser visto na Figura (1.3), nele, pode-se abrir qualquer arquivo de extensão .fits e o conteúdo de seu cabeçalho é mostrado na tela para o usuário.

FITS (*Flexible Image Transport System*) é um formato de arquivo digital para armazenamento, transmissão e análise de imagens científicas. Amplamente utilizado em astronomia, foi desenvolvido para dados científicos, portanto inclui várias propriedades úteis, como um cabeçalho, facilitando a manipulação pelos usuários.

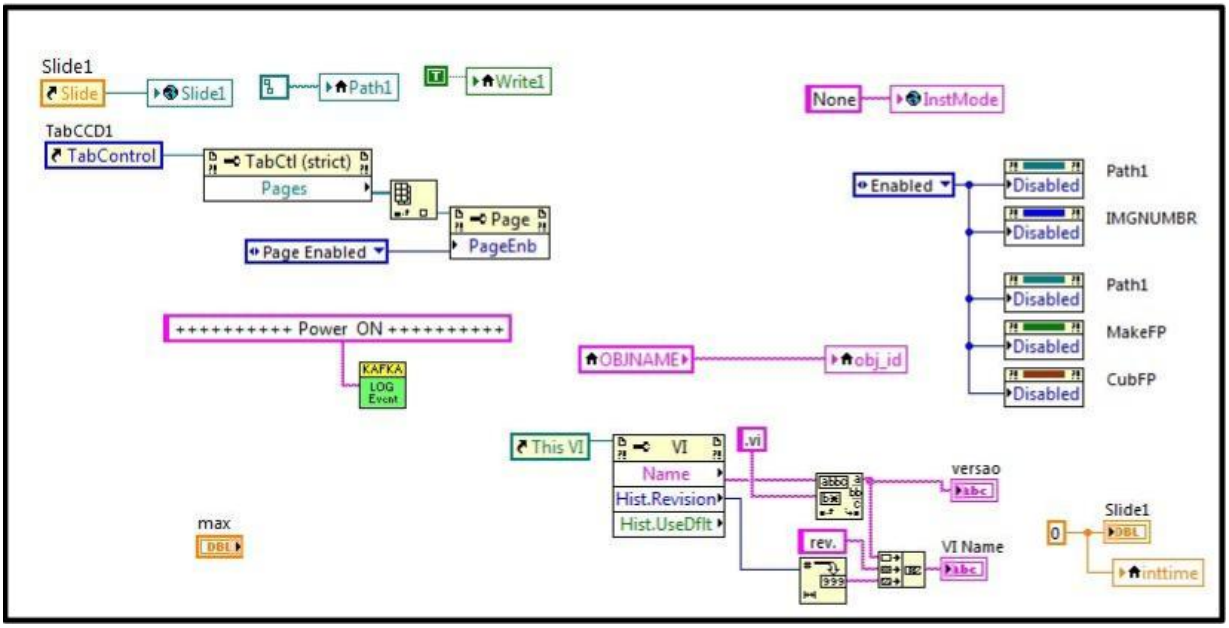

Figura 1.2: Exemplo da Interface de blocos do LabVIEW

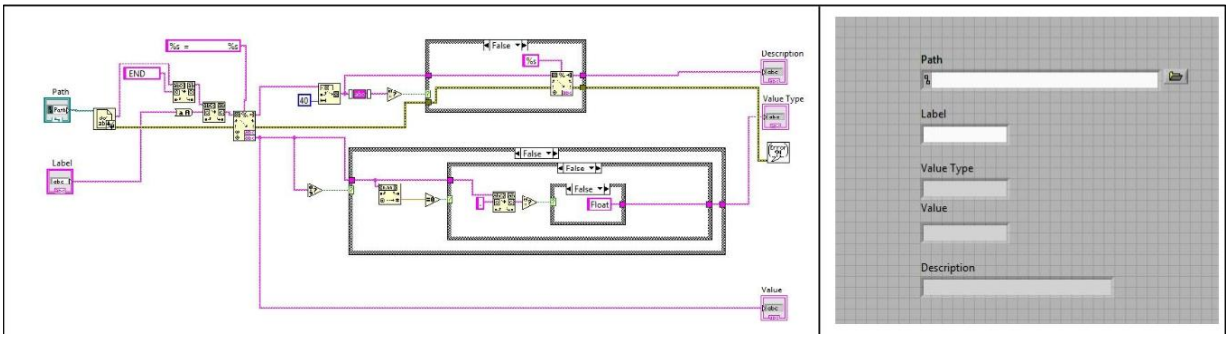

Figura 1.3: Esquema ilustrando um dos primeiros programas elaborados para leitura de dados do BTFI.

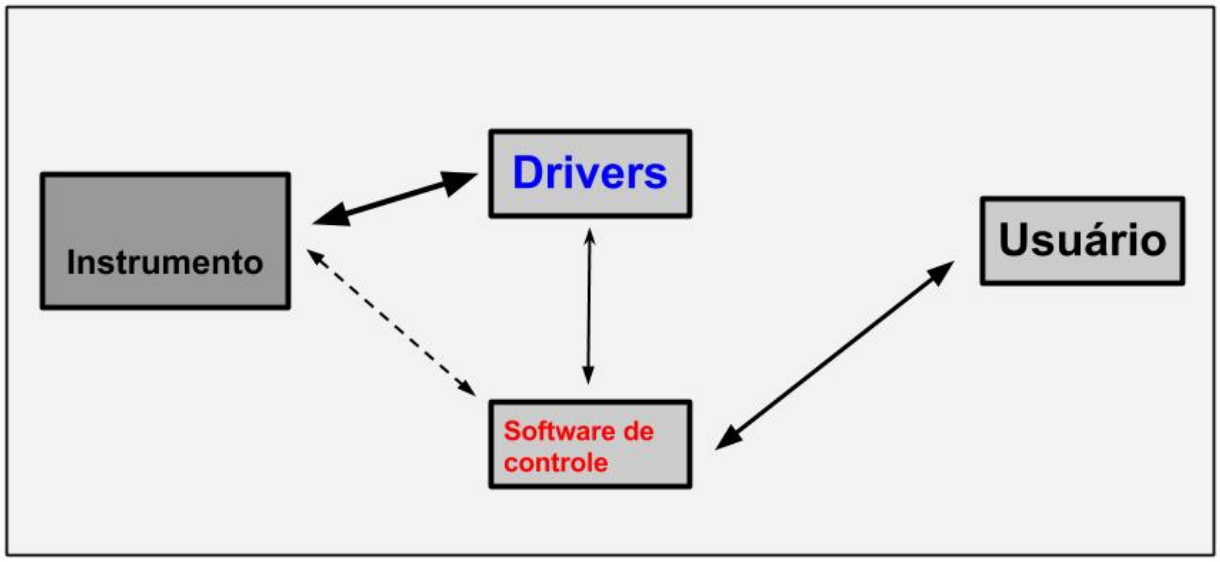

Figura 1.4: Diagrama exemplificando a comunicação entre o usuário e o instrumento

1.3 O Software de Controle - Kafka

Através do LabVIEW, foi desenvolvido o Software de Controle do instrumento. Com ele, o usuário consegue operar o instrumento através de comandos que são interpretados pelo programa e enviando para o instrumento. Um esquema básico ilustrando o funcionamento do software de controle é exemplificado na Figura 1.4.

Não existe uma comunicação direta entre o usuário e o instrumento, isto é inviabilizado pois a linguagem de máquina é a de mais baixo nível, na qual responde a comandos simples de texto contendo palavras específicas e números. Ou seja, é impossível para um usuário saber todos os comandos existentes e quando usá-los, sendo necessário alguma ferramenta que faça este intermédio entre as duas partes.

Com isto, existe o software de controle, um programa que capta as informações enviadas pelo usuário e transforme em algo que o instrumento possa entender. Da mesma forma, os dados gerados pelo instrumento são retornados para o usuário.

O processo de desenvolvimento deste programa, é algo que passa por várias etapas e necessita de muitas versões até que esteja completamente funcional e sem erros ou falhas. Além disso, conforme ele é usado, os usuários podem sugerir modificações para uma melhor eficiência do mesmo. O Kafka, é o nome dado a este software de controle do BTFI, no qual desde que o aluno entrou no projeto em Julho de 2012, teve mais de 30 versões distintas, cada uma mais aprimorada que a outra.

Figura 1.5: Janela principal do software de controle do instrumento, denominado Kafka.

Conforme indicado na Figura (1.5), pode-se observar que a janela principal de controle do instrumento possui inúmeras funções distintas que foram implementadas ao longo do tempo.

No total, o projeto conta com mais de 50 partições do código, cada uma responsável por executar alguma tarefa, sendo o Kafka a junção de todos estes códigos, permitindo o funcionamento completo do BTFI para o usuário.

Capítulo 2

Desenvolvimento

2.1 O Cabeçalho

A cada observação, quando imagens de algum objeto são adquiridas, certas informações são extremamente necessárias para que fiquem armazenadas no arquivo (normalmente de extensão .fits), afim de que quando estas forem tratadas, possam ser interpretadas de uma maneira correta.

Este campo onde as informações ficam contidas, é conhecido como *Header* (cabeçalho) e contêm detalhes a respeito da imagem, tais como coordenadas, escala, tamanho em pixels etc. Além disso, fornece outros dados como o nome de quem realizou a observação, o local, a data, o horário, etc.

No início do projeto, o Kafka possuia um cabeçalho simples, contendo não mais que 10 campos. Muitas informações importantes ficavam omissas, dificultando o processo de redução de dados. Em princípio, o trabalho consistiu de levantar os campos importantes faltantes, organiza-los de uma maneira que facilitasse a análise das imagens e que tornasse o processo de redução de dados algo mais prático. Dentre os campos acrescentados, estão o nome do observador, o objeto observado, as informações do observatório.

Durante a implementação dos novos campos, percebeu-se que o mecanismo antigo de organização do cabeçalho estava pouco eficaz, requisitando bastante tempo de programação do código para deixar na ordem desejada. A solução encontrada foi estabelecer um template (modelo), em que o operador pode organizar os campos em qualquer editor de texto e coloca-los na ordem que melhor desejar, eliminando a necessidade de alterar o código fonte do programa para realizar a mesma tarefa.

A Figura 2.1 mostra um exemplo de cabeçalho de imagem FITS e a figura contendo o

```
Initial Information
OBSERVER - "0 "              - IThe name(s) of the observer(s)

OBSERVAT - "0 "              - Name of the observatory
TELESCOP - "0 "              - Name of the telescope
INSTRUME - "0 "              - Instrument used to acquire data
MODE - "0 "                - Instrument mode
EQUINOX - "00,000 "                - Equinox of coordinates
OBJNAME - "0 "             - The name of the observed object
OBJTYPE - "0 "             - Type of observation
RA - "00,000 "             - Object's right ascention [hh:mm:ss.ss]
DEC - "00,000 "             - Object's declination [hh:mm:ss.ss]

Telescope information
DATE-OBS - "0 "              - [YYYY-MM-DDThh:mmss.ss]
DATE - "0 "              - YYYY-MM-DD
TIME - "0 "              - Start & Stop of Exposure
LONGITUD - "0 "              - Observatory's longitude [dd.dd]
LT - "0 "              - Local Time [hh:mm:ss.ss]
UTC - "0 "              - Coordinated Univ. Time [hh:mm:ss.ss]
LST - "0 "              - Local Sideral Time [hh:mm:ss.ss]
LATITUDE - "0 "              - Observatory's latitude [dd.dd]
Camera Information
EXPTIME - "0 "              - To exposure time for the current frame [seconds]
UTSHUT - "0 "              - Universal time when shutter opens.
iBTF Information
NFRAME - "0 "              - The number of frames that the current cube will have [--]
ANGSTEP - "0 "              - The iBTF angular displacement between two frames [degrees]
ANGSTART - "0 "              - The iBTF angular position in the firts frame [degrees]
ANGSTOP - "0 "              - The iBTF angular position in the last frame [degrees]
ANGNOW - "0 "              - The iBTF angular position for the current frame [degrees]
FRAMENOW - "0 "              - The number of the current frame inside the cube [--]
VPH1NAME - "0 "              - The name of the used VPH in the first support
VPH2NAME - "0 "              - The name of the used VPH in the second support
Fabry-Pérot Information
NFRAME - "0 "              - The number of frames that current cube will have [--]
GAPSTEP - "0 "              - The FP gap increment between two frames [microns]
GAPSTART - "0 "              - The FP gap size position in the first frame [microns]
GAPSTOP - "0 "              - The FP gap size position in the last frame [microns]
GAPNOW - "0 "              - The FP gap size position in the current frame [microns]
FRAMENOW - "0 "              - The number of the frame inside the cube [--]
General Information
MSKFLTR - "0 "              - Name of the filter used in the BTFI's Mask Wheel
IMGFLTR - "0 "              - Name of the filter used in the BTFI's Image Wheel
PUPFLTR - "0 "              - Name of the filter used in the BTFI's Pupil Wheel
```

Figura 2.1: Exemplo de um cabeçalho contido na imagem .fits.

mesmo completo encontra-se no apêndice A.

2.2 Melhorias na Eficiência

Conseguir noites para observar em um telescópio não é simples, portanto o tempo que se passa utilizando do instrumento tem que ser bem aproveitado. O ideal é adquirir a maior quantidade de dados com boa qualidade no menor tempo possível.

O iBTF estava com um problema em sua eficiência na hora de obter um cubo de dados e uma parte do presente trabalho foi de encontrar onde o processo estava lento e tentar solucioná-lo. Para isto, necessitou-se estudar o mecanismo de montagem desse cubo.

O processo de capturar imagens de um mesmo objeto em diferentes comprimentos de onda e unificá-las é conhecido como Cubo de Dados, por possuir informações tridimencionais do objeto (eixos x e y para as coordenadas e o z para os comprimentos de onda).

Uma das hipóteses levantadas foi que havia um pequeno atraso entre os motores se posicionarem corretamente e o início da acquisição de imagens, um atraso entre o usuário acionar o comando de iniciar a varredura (cujo termo em inglês *scan* será utilizado daqui

para frente) e o sistema processar e enviar para o instrumento.

Surgiu a oportunidade de o aluno ir até o Chile, e realizar um estágio nos laboratórios onde o iBTF estava separado do BTFI para alguns testes, afim de investigar melhor o problema. Contando com a colaboração do estudante de doutorado Bruno Quint, várias experiências foram realizadas.

O Controlador Lógico Programável (conhecido como CLP), é uma espécie de computador que como o próprio nome já diz, pode ser programado para realizar determinadas tarefas. Escrito em uma linguagem de programação de nível baixo, o CLP é utilizado, no caso do iBTF, para fazer a comunicação entre o computador e os motores. Ele faz a conversão do sinal de entrada que é serial, para impulsos elétricos que controlam os motores.

Após diversas modificações do Kafka, conclui-se que o problema estava localizado no driver que controlava os dois motores do instrumento, o qual estava programado na linguagem de Visual Basic 6.0, sendo necessário todo um processo de aprendizado e estudo do código (visto que ninguém estava experiente com esta linguagem).

O protocolo de comunicação entre o computador que controla o iBTF e o computador que hospeda o Kafka é chamado de TCP /IP , empregado para conexões remotas.

Ao analisar o código do driver dos motores, constatou-se que o mesmo apresentava uma função denominada "Pausa", na qual quando executada, fazia com que a leitura do programa ficasse em segundo plano e outras tarefas requisitando o processador fossem executadas em prioridade. Para verificar se esta era a causa do problema, removeu-se esta função por completo do código fonte e fez-se um teste do instrumento.

Ao fim do procedimento, não houve uma melhora significativa no desempenho, portanto a nova hipótese foi que havia um problema no código do programa em si. Além de que a comunicação via TCP parou de funcionar. Uma nova abordagem seria refazer o programa que controla o iBTF em outra linguagem, como LabVIEW, porém é algo extremamente trabalhoso (sendo necessário alguns meses de trabalho) e requer uma ampla experiência na área.

Começou-se a desenvolver o código em Visual Studio 10.0 (uma versão mais recente do software atual, que é 6.0), porém seria inviável com o tempo disponível.

Após alguns testes, diminui-se o tempo de pausa no código fonte do programa, além de aumentar a velocidade de comunicação entre o CLP e os motores. Conseguiu-se baixar

Tabela 2.1 - Tabela com os dados obtidos dos tempos de exposição em um scan

Imagens por Posição	Coeficiente Angular	Coeficiente Linear
1	1	2,17(2)
2	1	2,91(2)
4	1	4,40(1)
5	1	5,10(1)

o tempo que o iBTF demorava para realizar um movimento de 11 segundos para 3 segundos, porém esta melhoria refletiu em alguns erros na sincronia das câmeras, os quais não apareciam quando o equipamento estava lento.

Após corrigir o problema de sincronia das câmeras, com o novo programa funcionando corretamente, um scan de 400 quadros, com 2 exposições de 0,5s, com uma variação ângular de 0,05º, em que antes demorava 1 hora e 40 minutos, foi concluido em 40 minutos.

Afim de melhorar a estabilidade, fez-se um tratamento de erros do Kafka, que, quando constatado um problema durante um scan, o usuário tivesse a opção de continuar da onde parou ou cancelar o procedimento. Essa opção elimina a necessiade de ter que recomeçar uma tarefa desde o início, que demandaria muito tempo.

Acrescentou-se ao Kafka, na janela de inicialização, o endereço de IP e portas do iBTF, assim o usuário pode verificar se estes estão corretos.

Com o iBTF mais rápido, investigou-se a velocidade de comunicação das câmeras. Para isto, fez-se um scan com vários tempos de exposição e, o tempo perdido nas câmeras foi calculado (através de uma VI do LabVIEW). Fez-se um gráfico com os dados que pode ser visto na Figura 2.2.

Fez-se uma comparação entre o tempo de exposição ideal e o tempo de exposição real. Para uma imagem por posição, verificou-se que seu coeficiente linear deu 2,17(2) segundos, ou seja, o tempo que a câmera demora em responder a um comando para exposição de 0s. Seu coeficiente angular, assim como todas as outras retas, foi de 1,0s, ou seja, as câmeras não estão perdendo tempo conforme o tempo de exposição aumenta, é algo constante.

Sabendo deste detalhe, fez-se o seguinte cálculo: O tempo de perda por exposição será igual ao Tempo Real menos o Tempo de resposta de uma exposição de 0s (calculado acima como 2,17s) dividido por n-1 imagens na mesma posição. Este cálculo retorna os valores que se encontram nas Tabelas 2.1 e 2.2 respectivamente.

Ou seja, em um scan, o tempo perdido pelas câmeras por posição será de aproximada-

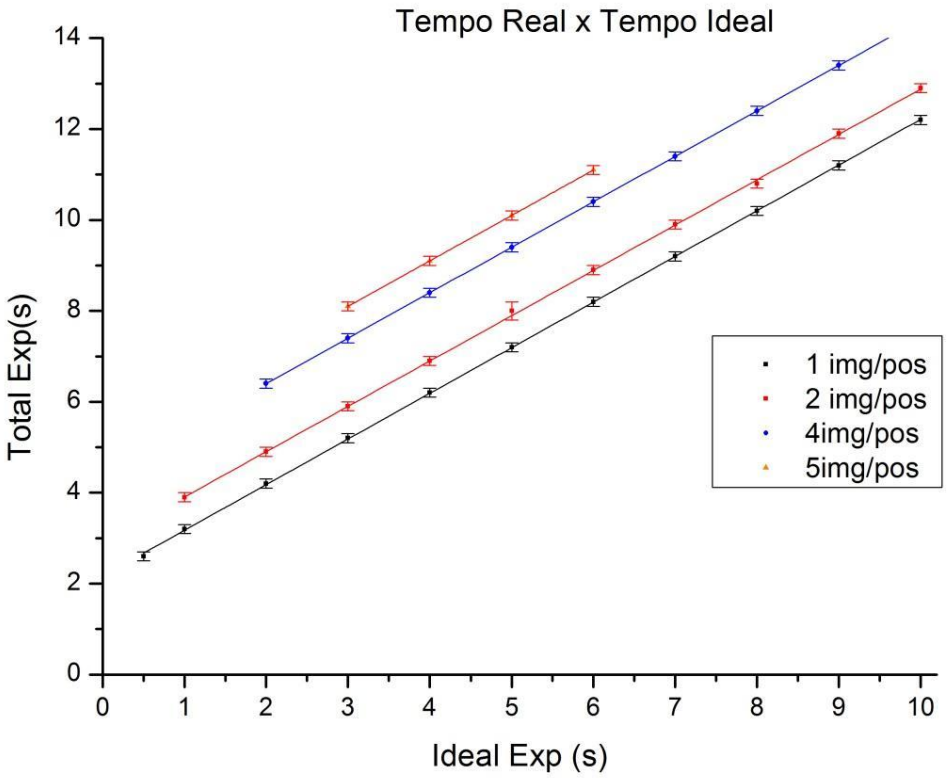

Figura 2.2: Figura comparando o tempo real que se gasta para fazer cada imagem versus o tempo ideal.

Tabela 2.2 - Tabela contendo o tempo perdido para cada configuração de scan.

Imagens por Posição	Coeficiente Linear	Tempo Perdido por Exposição
1	2,17(2)	0
2	2,91(2)	0,74(1)
4	4,40(1)	0,75(1)
5	5,10(1)	0,74(1)

mente $2,17 + 0,74 * (n - 1)$ segundos por posição, onde n é o número de imagens.

Este valor de 0,74s é aceitável pois as câmeras possuem um tempo necessário para realizar suas operações de rotina entre uma exposição e outra.

Conclui-se que era necessário investigar o porquê da demora de 2,17 segundos para começar a obter a primeira imagem de cada posição, visto que em um scan de 400 passos, gera um atraso de 14,47 minutos. Caso seja necessário reescrever o programa de controle do iBTF, possuímos as ferramentas necessárias para desenvolve-lo em LabVIEW utilizando o protocolo OPC Server.

No fim do estágio no Chile, foi possível testar no telescópio a calibração e montagem do Fabry-Perrot, afim de entender melhor o instrumento e seu funcionamento.

2.3 TCS, Roda de Filtros

Conforme ocorre a observação, algumas informações são importantes para que o usuário tenha conhecimento, como por exemplo a umidade, a velocidade do vento, a posição em que se encontra o telescópio, dentre outros campos que ser referem às condições observacionais. Estes dados estão contidos no computador do observatório, sendo necessário obtê-los de alguma forma e disponibilizar para quem estiver usando o BTFI. A ferramenta utilizada para isto, é chamada de TCS (*Telescope Control Systems*), um dos objetivos do presente trabalho foi de implementá-la no Kafka. Como pode ser observado na Figura 2.3, todas as informações obtidas pelo observatório mostradas para o usuário.

O BTFI possui rodas de filtros e uma roda de máscara, que podem ser utilizados durante a observação. Porém, quando um determinado filtro ou máscara era selecionado, este não constava corretamente no cabeçalho, sendo extremamente difícil depois identificar qual foi utilizado.

Afim de resolver este problema, criou-se um mecanismo no software que conseguia detectar qual filtro está ativo no momento da obtenção da imagem, para que esse dado fosse armazenado no sistema e depois colocado no cabeçalho corretamente.

Esta implementação só foi possível pois o Kafka permite que o usuário configure quais filtros e máscaras utilizará durante a observação, selecionando as opções desejadas e salvando no sistema, guardando a mesma configuração para observações futuras.

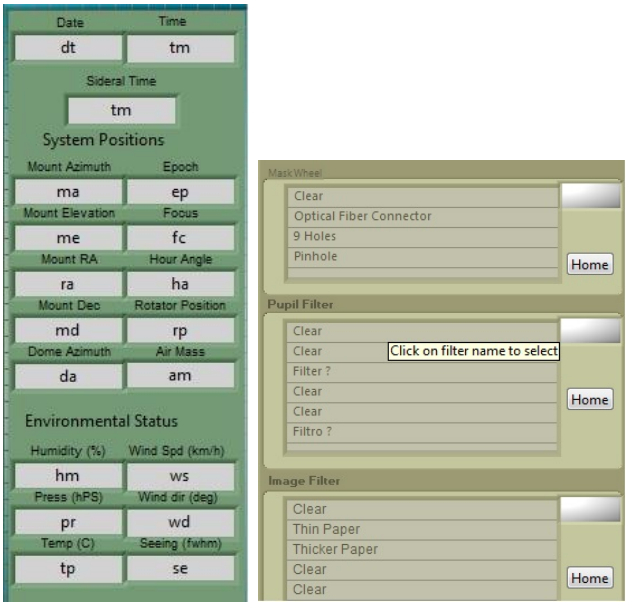

Figura 2.3: Ilustração das informações do TCS (à direita) e sobre as rodas de filtro e máscara (à esquerda).

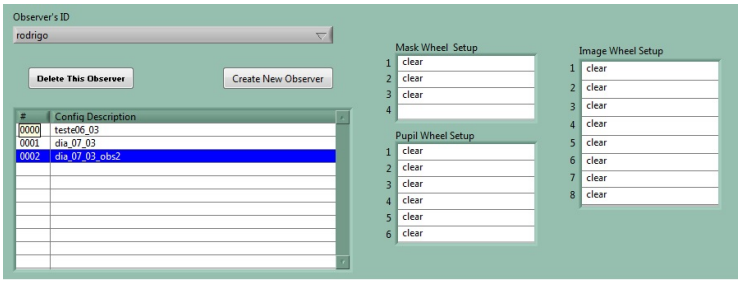

Figura 2.4: Apresentação das configurações das máscaras e filtros estabelecidas por um determinado usuário.

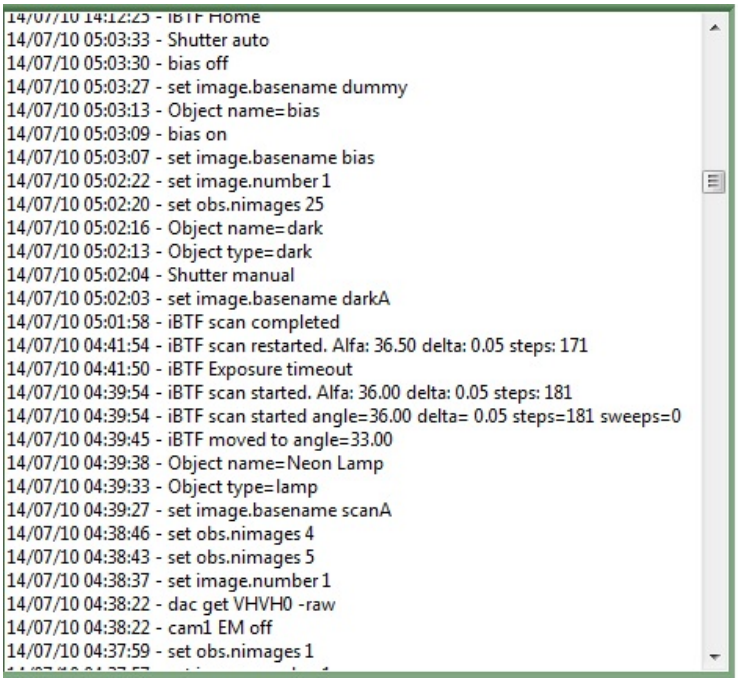

Figura 2.5: O Log gerado pelo programa, da ordem dos eventos mais recentes para os mais antigos.

2.4 Log do Sistema

Um programa bem desenvolvido possui um sistema de armazenamento de todas as ações realizadas, com o respectivo horário de execução. Este mecanismo de captação de informações é conhecido como Log e possui como principal finalidade poder manter um histórico de tudo que aconteceu durante o funcionamento do programa.

Caso ocorra algum erro, ou seja necessário recaptular o que foi utilizado durante determinado horário e dia de observação, basta consultar o log do sistema e procurar pela data desejada. Para que o Log sempre consiga registrar todas as ações, é necessário que se mantenha atualizado no Software, pois quando alguma parte nova do código é implementada, esta não fica registrada, sendo necessário configurar manualmente.

Uma grande alteração realizada foi a de inverter a ordem em que o Log é mostrado para o usuário. Anteriormente a sequência era dos eventos mais antigos para os mais recentes, dificultando a obtenção das informações a respeito do que estava acontecendo no instante de observações. Agora mostra-se em uma ordem contrária, dos mais atuais para os mais velhos.

2.5 O Emprego do Controle de Versão no Kafka

O Controle de Versão é algo extremamente importante quando se refere a projetos que estão em constante desenvolvimento, por várias pessoas distintas simultaneamente.

O Kafka atualmente está passando por um processo de aperfeiçoamento, significando que várias versões estão sendo testadas ao longo do tempo. Afim de manter uma documentação e facilitar a organização destas versões, o emprego do Controle de Versão tornou-se necessário.

O LabVIEW, até a versão 8.5, possuía seu próprio controle de versão integrado ao software, porém devido a alguns problemas encontrados, ele foi removido nas versões posteriores.

Atualmente há uma gama de opções fornecidas por software de terceiros que oferecem integração com o LabVIEW.

O escolhido por oferecer um suporte multiplataforma (visto que utilizamos tanto Windows quanto Linux no projeto) foi o Git. Este pode ser tanto operado por um terminal, com linhas de comando, ou com um programa de interface gráfica, atendendo qualquer tipo de usuário que prefira uma das duas opções.

Além do Git, foi necessário configurar um servidor para que pudesse armazenar as versões dos programas e que fosse de fácil utilização para os usuários, no caso, escolheu-se o projeto chamado Bitbucket .

O conceito de o software de controle é proporcionar um desenvolvimento de um código de programação por uma equipe de pessoas, em que possibilite uma melhor organização dos desenvolvimentos e melhorias do mesmo.

O Git utiliza-se da filosofia de “Branches” e “Mergers”, conforme está ilustrado na Figura 2.5.

Para cada alteração no código, o programador necessita fazer uma ação chamada “Commit”, nela, é obrigatório inserir um comentário a respeito da alteração realizada, além de ficar armazenado o autor e a data. Com isto, fica muito mais fácil identificar cada versão e quais aspectos foram alterados.

A opção de “Merge”, que pode ser realizada apenas com a autorização de um administrador do projeto (evitando acidentes ou versões problemáticas), permite que uma parte alheia ao código seja desenvolvida em separado e depois fundida ao projeto, sem ser

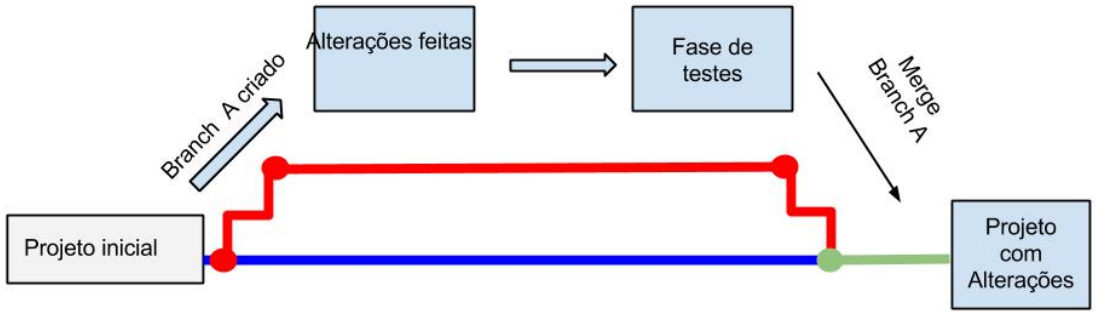

Figura 2.6: Fluxograma indicando o funcionamento do controle de versão empregando o conceito de "Mergers"e "Branches".

necessária a paralisação total do mesmo.

Capítulo 3

Conclusões

O trabalho começou com o aprendizado de uma linguagem de programação e o estudo do funcionamento do instrumento no primeiro semestre, depois passou por um processo de melhorias e alterações nos seguintes, até chegar em uma fase atual de estabilidade.

O Kafka encontra-se em um patamar de aprimoramentos, já não apresenta tantos erros quanto apresentava nas versões anteriores. O usuário pode utilizá-lo por várias horas sem se deparar com algum erro grave que o impeça de continuar suas observações, mas como qualquer programa, sempre haverá melhorias que poderão ser implementadas.

Diversas tarefas foram cumpridas no presente trabalho para que isto fosse possível. A nova eficiência, por exemplo, um número maior de dados pode ser produzido em um menor tempo, tanto para o iBTF quanto para o FP, aproveitando melhor uma noite de observação no telescópio. O cabeçalho está mais completo, facilitando o processo de redução dos dados. Há também a possiblidade de caso ocorra algum erro durante a produção de um cubo de dados, este continuar o procedimento à partir do ponto em que parou.

Um aspecto importante que foi levantando durante o desenvolvimento, é a organização necessária do projeto e suas novas versões. É preciso todo um histórico de cada alteração feita, e se necessário, sempre ter um ponto de retorno, caso haja alguma modificação problemática. Isto foi muito beneficiado pelo emprego do Controle de Versão. O projeto passou a ter todo um histórico de modificações feitas de uma maneira automatizada e acessível a qualquer programador do grupo, podendo verificar a diferença entre versões, realizar backups dentre outras tarefas do gênero.

Para o futuro, há a proposta de se modificar de modificar o arquivo de Log do programa de uma maneira que este fique mais organizado, separando os documentos do relatório de funcionamento por data e limitando o tamanho dos mesmos, facilitando a identificação de

problemas e ações realizadas. Outro objetivo é o de aplicar o ajuste do foco do telescópio pelo próprio Kafka, permitindo uma alteração pelo próprio aplicativo, sem a necessidade do operador do telescópio ter que intervir. É um trabalho interessante pois empregará além da linguagem do LabVIEW, scripts de Python de análise de imagens, para saber quando o foco estará mais otimizado.

A eficiência do instrumento ainda pode ser melhorada, fazendo mais algumas alterações no programa e modificando o funcionamento do scan, mas isto ficou para ser discutido futuramente pois é necessário toda uma remodulação do esquema de obtenção de imagens do software.

Outro trabalho que poderia ser realizado futuramente, é o de alterar o driver dos motores do iBTF, visto que estão programados em uma linguagem obsoleta (Visual Basic 6.0), mas requer um tempo de dedicação muito alto e seria necessário um estudo maior para decidir se os benefícios compensam todo os recursos gastos nessa modificação.

Referências Bibliográficas

Bitbucket Página do projeto do Bitbucket `https://bitbucket.org/`

BTFI Página do projeto do BTFI `http://www.astro.iag.usp.br/~btfi//`

Git Página oficial do projeto Git `http://git-scm.com/`

LSI Página da Usp para a obtenção do LabVIEW, acessível apenas dentro da Universidade `http://www.lsi.usp.br/labview/`

NationalInstruments Página oficial do LabVIEW `http://brasil.ni.com/`

Taylor K., The Brazilian tunable filter imager for SOAR, SPIE Proceedingsl, 2010, vol. 7739

TCP P., Informações sobre o protocolo de comunicação TCP. `http://en.wikipedia.org/wiki/Transmission_Control_Protocol`

Apêndice

Apêndice A

Cabeçalho Atual

```
COMMENT0        Initial Information     -----
OBSERVAT        CTIO    Name of the observatory
TELESCOP        SOAR    Name of the telescope
INSTRUME        BTFI    Instrument used to acquire data
OBSERVER                The name(s) of the observer(s)
INSTRMOD        None    Instrument mode (FP, iBTF or None)
OBJTYPE         Type of observation (flat, object, ...)
OBJNAME                 The name of the observed object
DATE            dt      Date
TIME            tm      Time
LATITUDE        -30,740 Observatory's latitude [dd.dd]
LONGITUD        -70,440 Observatory's longitude [dd.dd]
COMMENT1        Telescope Information   ----
MOUNTRA         ra      RA
MOUNTDEC        md      Mount Dec
MAZIMUTH        ma      Mount Azimuth
MELEVAT         me      Mount Elevation
HOURANGL        ha      Hour Angle
SIDTIME         tm      Sideral Time
COMMENT2        Camera Information      ----
INTGTIME        0,000   Exposure time (ms)
IMGNUMBR        0       Number of current image
IMGPPOS         0       Number of images to be taken for each position
COMMENT3        iBTF Information        ----
IBTFMODE        HOME    iBTF Operation Mode
TFAINIT         25,000  Inicial angle on iBTF scan
TFADELT         0,010   Delta angle on iBTF scan
IBNUMSTP        7       Number of steps in iBTF scan
IBCURSTP        0       Current step in iBTF scan
IBCURANG        0,000   iBTF Current Angle
COMMENT4        Fabry-Perot Information ----
FPZINIT         0       Initial position in CS100 scan
FPZDELT         0       Delta step in CS100 scan
FPZSTEP         0       Number of steps in CS100 scan
FPZCURR         0       Current positon in FP scan
FPSWENUM        1       Number of sweeps in CS100 scan
FPSWENUM        1       Number of sweeps in CS100 scan
COMMENT5        General Information     ----
MaskLB          none    Mask name or model
PupLB           none    Pupil filter name or model
ImgLB           none    Image Filter name or model
VPH1NAME                The name of the used VPH in the first support
VPH2NAME                The name of the used VPH in the second support
SHUTMODE        (STRING)        MANUAL  Shutter Operation Mode
SHUTSTAT
None            None    None
DOMEAZIM        da      Dome Azimuth
SHUTTER         sh      Shutter
ROTNAME         rn      Rotator Name
ROTPOSIT        rp      Rotator Position
AIRMASS         am      Air Mass
FOCUS           fc      Focus
EPOCH           ep      Epoch
PRESSURE        pr      Press (hPS)
TEMPC           tp      Temp (C)
DEWPOINT        dw      Dewpoint (C)
WINDSPED        ws      Wind Spd (km/h)
WINDDIR         wd      Wind dir (deg)
SEEING          se      Seeing (fwhm)
EM1             False   EM gain Camera 1
EM2             False   EM gain Camera 2
HUMIDITY
```

Figura A.1: Cabeçalho atual completo gerado pelo software.