UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTENSIÓN DE FILTRO DE KALMAN DE APROXIMACIÓN NO LINEAL PARA LA DETECCIÓN DE OBJETOS ASTRONÓMICOS

TESIS PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

PALOMA PÉREZ GARCÍA

PROFESOR GUÍA:
SR. PABLO ESTEVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
SR. BENJAMÍN BUSTOS
SR. ZZZ ZZZ ZZZ
SR. ZZ ZZ ZZ
SR. ZZZ ZZZ ZZZ

SANTIAGO DE CHILE
DICIEMBRE 2018

Resumen Ejecutivo

...

Agradecimientos

Paloma Pérez García

Índice general

Resumen Ejecutivo I

Agradecimientos III

1. Introducción 1

1.1. Motivación . . . 1

1.2. Organización de la tesis . . . 2

2. Antecedentes 4

2.1. Supernova tipo II . . . 4

2.2. High Cadence Transient Survey: HiTS . . . 5

2.3. El filtro de Kalman . . . 6

2.3.1. Filtro de Kalman Básico . . . 7

2.3.2. Filtro de Kalman de Máxima Correntropía . . . 11

2.3.3. Filtro de Kalman Unscented . . . 12

2.4. Laboratorio nacional de computación de alto desempeño (NLHPC) . . . 15

3. ¿Qué se entiende por *linealidad*? 17

3.1. El filtro de Kalman para un sistema localmente lineal . . . 17

4. Desempeño del programa original 18

4.1. Tiempo de ejecución . . . 19

4.2. Uso de memoria . . . 20

5. Refactoring 24

5.1. Manejo de datos de entrada . . . 24

5.1.1. Lectura y preparación de imágenes . . . 26

5.2. Determinación de flujos . . . 27

5.3. Filtros originales . . . 27

5.3.1. Strategies de predicción . . . 28

5.3.2. Strategies de corrección . . . 28

5.3.3. Filtros refactorizados . . . 28
5.4. Detección de candidatos . . . 28
5.4.1. SourceFinder . . . 29
6. Nueva funcionalidad **32**
6.1. Manejo de la rutina: RoutineHandler . . . 32
6.2. Detección de fenómenos trasientes: **TPDetector** . . . 33
6.3. DataContent . . . 33
7. Resultados **35**
7.1. Desempeño . . . 35
7.1.1. Filtros básico y de máxima correntropía . . . 35
7.1.2. Filtro unscented . . . 35
7.2. Pruebas en Leftraru . . . 35
7.2.1. Filtro básico . . . 35
7.2.2. Filtro de máxima correntropía . . . 35
7.2.3. Filtro unscented . . . 35
8. Conclusiones **36**
Capítulo Adicional que no es apéndice **37**
Apéndices **38**
A . Glosario . . . 38
B . Refactoring . . . 38
B .1. Librerías usadas para el refactoring . . . 38
B .2. Archivo de entrada: configuración de paths . . . 39
C . Nueva funcionalidad . . . 39
C .1. Modelo de archivo de almacenamiento de resultados . . . 39
D . Unit-tests . . . 39
D .1. Refactoring . . . 39
Referencias **41**

Índice de cuadros

4.1. Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman Básico. 20

4.2. Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman Básico. 20

4.3. Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman de Máxima Correntropía. 20

4.4. Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman de Máxima Correntropía. 21

4.5. Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman Básico. 21

4.6. Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman de Máxima correntropía. 21

4.7. Memoria principal (en unidades de MB) usada durante la ejecución del programa original usando la versión básica del filtro de Kalman. 21

4.8. Memoria principal (en unidades de MB) usada durante la ejecución del programa original usando filtro de Kalman de máxima correntropía. 21

Índice de figuras

2.1. Representación simple y no a escala de la estructura de una estrella masiva previo a su explosión como supernova. Los elementos más pesados se alojan en el centro, mientras que los más livianos, como el hidrógeno o el helio lo hacen en la capa más externa. 4

2.2. Curvas típicas de supernovas I (enana blanca) y II (estrella masiva). 5

2.3. Diagrama que ilustra la evolución del brillo de una supernova en términos de luminosidad solar ($L_\odot$) durante días. Se resalta el fenómeno de shock breakout apenas comienza el incremento de la luminosidad de la supernova. Esta imagen fue publicada en la página de la NASA destacando la primera vez que un evento como este es *capturado* en la banda visible (por el telescopio espacial Kepler). 6

2.4. A la izquierda, estructura de la cámara DECam poblada con 62 chips CCDs. A la derecha, imagen *flat field* desde DECam. 7

2.5. Orientaciones sobre el cielo y la huella espacial del arreglo de detectores en el plano focal. Se destacan los CCD cuya etiqueta comienzan con S o N, ya que estos corresponden a los detectores encargados de obtener las imágenes científicas. El etiquetado de estas componentes puede ser engañoso debido a que las iniciales de norte (North) y sur (South) están invertidas en relación a la orientación del cielo. 8

2.6. Distribución espacial de los campos observados durante los primeros semestres de los años 2013 (gris), 2014 (azul) y 2015 (naranjo). En tono rojo, los mismos campos del año 2015 y 2014 (superposición). 8

2.7. Representación del proceso de predicción (obtención de cantidades a priori) de las cantidades $\hat{x}_{k|k-1}$ y $P_{k|k-1}$; y de corrección (estimación a posteriori) para obtener las cantidades $\hat{x}_{k+1|k}$ y $P_{k+1|k}$. 10

2.8. Representación del funcionamiento del filtro UKF. En esta oportunidad se hace uso de la función f y h para obtener las transformaciones $x_{k-1} \rightarrow x_k$ y $x_k \rightarrow z_k$. Esto se logra con la evaluación de de los 2N+1 puntos generados durante la etapa de predicción (y posteriormente en la etapa de corrección). 13

4.1. Diagrama de actividad del programa original. Se aprecían dos ciclos principales: el primero está destinado a la búsqueda de una supernova de HiTS, y el segundo a la revisión de la lista de posibles candidatos encontrados durante la verificación de la supernova de HiTS. Notar que hay pasos que se repiten en la realización de ambos análisis. 19

4.2. Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman Básico. . . . 22

4.3. Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman de correntropía máxima. 23

5.1. Familia de filtros de Kalman y patrón strategy usado en la implementación de los métodos `predict` y `correct`. 29

8.1. Ejemplo de distorsión de una fuente al aplicar un kernel de PSF específico. El resultado se observa en el cuado *Image*. 38

1 Introducción

1.1. Motivación

La astronomía es uno de los campos científicos que más se ha visto afectado por el rápido crecimiento en la generación de datos debido al fuerte desarrollo de nuevas tecnologías de la información y de nuevos instrumentos destinados a la observación. Este crecimiento ha gatillado un aumento importante en la demanda de una nueva generación de métodos que puedan procesar esta oleada de información o big data astronómico.

Ejemplos de proyectos que actualmente producen una gran cantidad de datos a través de telescopios en diferentes partes del mundo son: el Panoramic Survey Telescope and Rapid Response System (Pan-STARRS), el Visible and Infrarred Survey Telescope (VISTA), el VLT Survey Telescope (VST), el Dark Energy Camera Legacy Survey (DECaLS) y el Hyper Suprime-Cam Subaru Strategic Program (HSC SSP). Estos surveys están caracterizados por un amplio *étendue* (producto entre el área del espejo de un telescopio y su ángulo sólido proyectado en el cielo).

En el futuro, telescopios como el Large Synoptic Survey Telescope (LSST) [6] (que entrará en funcionamiento a mediados del 2022) continuarán revolucionando la era del big data en astronomía con etendues y cámaras CCD mucho más grandes de lo que hasta el día de hoy se utiliza. En particular se espera que el LSST produzca un número de alertas trasientes (avisos de objetos encontrados y su consecuente seguimiento) del orden del millón cada noche. La capacidad de detectar nuevos objetos de interés dependerá de la calidad de los datos y de los algoritmos de tiempo real destinados a generar las alertas mencionadas.

Actualmente existen sondeos como el High Cadence Transient Survey (HiTS) [3] cuya finalidad es la búsqueda de fenómenos trasientes rápidos con escalas de tiempo que van desde las horas a días, utilizando secuencias de observaciones de la cámara DECam del telescopio Blanco (en Cerro Tololo) para la detección y posterior reporte de objetos candidatos a supernova. En su trabajo proponen un método de detección de potenciales candidatos a supernova a través de la discriminación de píxeles que involucren un incremento trasiente de intensidad.

Esta discriminación comienza con la determinación del flujo a través de la intensidad de los píxeles que puedan corresponder a una estrella y a la variación de cada uno de ellos usando métodos iterativos de filtrado en secuencias de imágenes de largo arbitrario. Los filtros pensados para HiTS corresponden a miembros de una familia de filtros conocidos como *filtros de Kalman*. En particular se hace uso de los filtros de Kalman básico [5] y de correntropía máxima [2].

El reciente trabajo desarrollado por Pablo Huentelemu [8] (M.Sc.) guiado por Pablo Estévez (PhD) y Francisco Förster (PhD), propone el uso de estos filtros cuyos resultados de verdaderos positivos de supernovas conocidas se pretenden mejorar (hasta ahora alcanza alrededor de un 44 %), por lo que se plantea la posibilidad de diseñar algún otro criterio de filtrado y de estudiar los umbrales y parámetros del programa.

En el presente trabajo se pretende desarrollar un nuevo módulo de filtro de Kalman que sea robusto a sistemas no-lineales para la detección de potenciales candidatos a supernova, para posteriormente ser embebido en un programa que pueda funcionar en línea generando alarmas de detección. Por otro lado se busca implementar un software que contenga los métodos de filtrados originales (versiones clásica y de máxima correntropía del filtro de Kalman) a través de un refactoring del programa base. Paralelamente, se desea encontrar rangos apropiados para los umbrales requeridos para el filtrado.

1.2. Organización de la tesis

Este documento sigue los siguientes capítulos: en el capítulo 2 se describe en qué consiste una supernova, el proyecto HiTS, la relevancia de la linealidad de un sistema (y su impacto en la estadística del mismo en la aplicación de un método de filtrado como el filtro de Kalman) así como el background matemático de los métodos de filtrado usados en el trabajo original (filtros de Kalman básico y de máxima correntropía) y el filtro a implementar (filtro de Kalman unscented).

El siguiente capítulo, 4, se estudian los resultados obtenidos con el código original del programa sobre el cual se trabajará, estableciéndose una breve discusión sobre estos resultados.

Posteriormente, en el capítulo 5 se describe el refactoring del programa del código original: se enlistan nuevos métodos para el manejo de archivos y cambios realizados en la implementación de los métodos iniciales.

En el capítulo 6 se exponen los pasos del desarrollo del nuevo filtro así como nueva funcionalidad relacionada con la carga de resultados previamente guardados en pos de hacer este programa un

proceso on-line.

Posteriormente, en el capítulo 7.2.3, se describen los resultados obtenidos de las pruebas descritas en el capítulo anterior comparando tasas de verdaderos positivos y falsos negativos, así como analizando las estadísticas de los grupos de píxeles candidatos (tanto descartados como confirmados).

Finalmente las conclusiones, contenidas en el capítulo 8, resumen el aprendizaje obtenido durante este trabajo así como los resultados relevantes obtenidos durante las pruebas establecidas. Además se plantean nuevas líneas de trabajo futuro.

2 | Antecedentes

2.1. Supernova tipo II

Una supernova de tipo II corresponde a un evento estelar con el que finaliza una estrella masiva (aquellas que en su proceso de formación poseen una masa superior a 10 masas solares [1]). Al ya no contar con combustible necesario para llevar a cabo reacciones nucleares que puedan contrarrestar su propia gravedad (su núcleo ya no puede formar elementos más pesados que el hierro o el níquel), y si la presión degenerada de los electrones del plasma de la estrella no es suficiente para soportar este peso, la estrella se contrae abruptamente incrementando el peso de su centro a 10^{10} K.

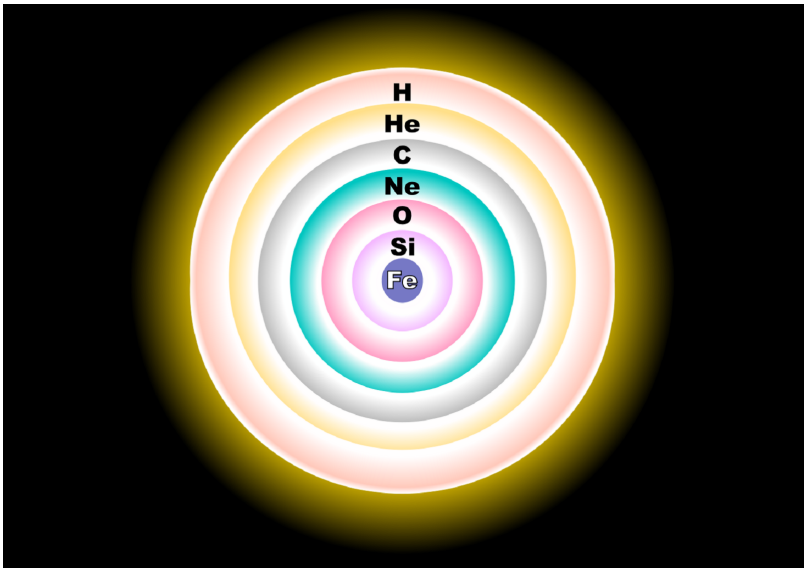


Figura 2.1: Representación simple y no a escala de la estructura de una estrella masiva previo a su explosión como supernova. Los elementos más pesados se alojan en el centro, mientras que los más livianos, como el hidrógeno o el helio lo hacen en la capa más externa.

Debido al aumento de temperatura, los electrones del núcleo adquieren energía cinética suficiente para escapar, desapareciendo así la presión que ejercían hacia el exterior. Finalmente el núcleo colapsa liberando energía gravitacional y con ella las más externas de la estrella son expulsadas en una gran explosión. Este fenómeno se denomina *supernova* e implica un aumento repentino del

[1] $M_\odot = (1.98847 \pm 0.00007) \times 10^{30}$ Kg

brillo de una estrella, incrementando su brillo en un factor de 10^8 veces, pudiendo incluso ser más brillante que la galaxia que la alberga.

En general la variación de la luminosidad de una supernova corresponde a una curva que crece rápidamente los primeros días (u horas), alcanzando un máximo, para luego decaer. Cabe destacar que existe otro tipo de supernova, las de tipo I y corresponden a otro fenómeno en donde participa una clase de estrella denominada enana blanca junto a otra estrella de cualquier otro tipo. La primera, al poseer una gravedad tan alta en su superficie es capaz de tomar material de su compañera con lo que al superar las 1.44 $M_\odot$ se desencadenaría la explosión de supernova I.

Las supernovas de tipo I presentan un decaimiento casi continuo una vez alcanzado el máximo, mientras que las de tipo II presentan dos caídas: una inmediatamente después de su máximo y otra una vez finalizado un periodo de decaimiento suavizado (figura 2.2). Otra forma de diferenciarlas es la presencia de trazas de hidrógeno en los espectros de estas: las supernovas de tipo I practicamente no presentan hidrógeno (líneas de absorción distintivas) a diferencia de las de tipo II.

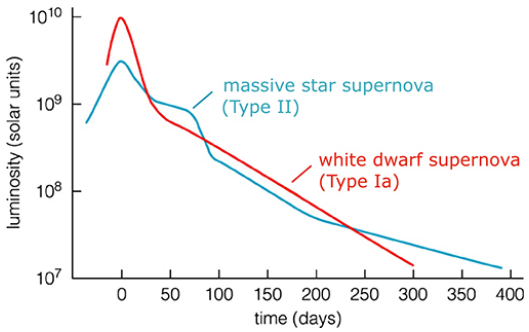


Figura 2.2: Curvas típicas de supernovas I (enana blanca) y II (estrella masiva).

2.2. High Cadence Transient Survey: HiTS

El High Cadence Transient Survey [3] (desde ahora, HiTS) es un survey cuyo objetivo principal es el detectar y seguir fenómenos transientes estelares en escalas de tiempo que van desde horas a días, con especial atención a fases tempranas de explosiones de supernovas (primeras horas): el objetivo original de HiTS corresponde a la detección de un fenómeno llamado *shock breakout* (SBO), un fenómeno que ocurre inmediatamente después del colapso del núcleo de una estrella roja supergigante (una de las posibles etapas finales de una estrella masiva antes de *explotar* en supernova II). Ver figura 2.3[2].

[2] $L_\odot = 3.828 \times 10^{26}$ W

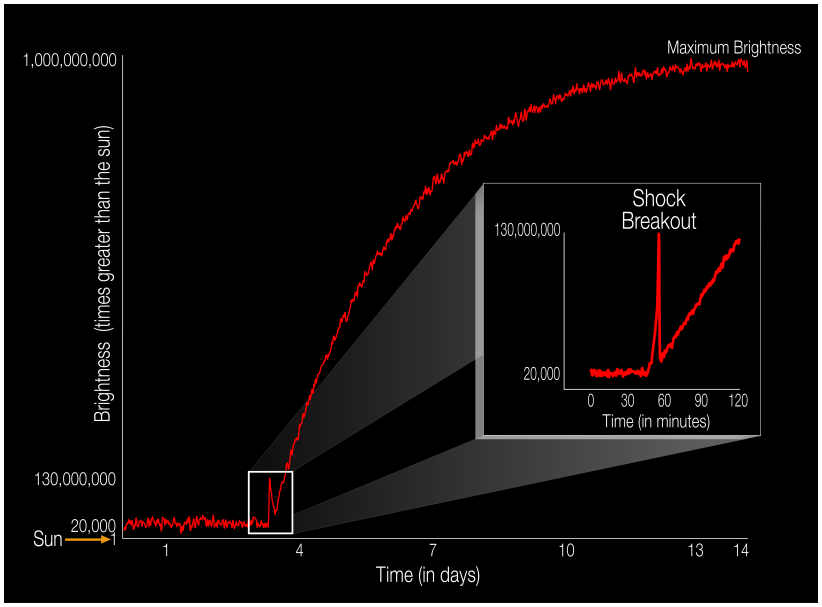


Figura 2.3: Diagrama que ilustra la evolución del brillo de una supernova en términos de luminosidad solar ($L_\odot$) durante días. Se resalta el fenómeno de shock breakout apenas comienza el incremento de la luminosidad de la supernova. Esta imagen fue publicada en la página de la NASA destacando la primera vez que un evento como este es *capturado* en la banda visible (por el telescopio espacial Kepler).

HiTS utiliza la Dark Energy Camera (DECam, figura 2.4) para la obtención de sus imágenes. Esta cámara se encuentra montada en el Telescopio Blanco del Observatorio de Cerro Tololo (CTIO) en la región de Coquimbo, Chile. Esta cámara posee 62 detectores CCD de 2048×4096 píxeles para la obtención de imágenes científicas y otros 12 para la guía, alineamiento y enfoque.

Durante este proyecto se han realizado tres campañas de observación en los años 2013, 2014 y 2015 durante el primer semestre de cada año. Se escogieron 40 campos y 4 épocas por noche (también por campo) para observaciones para el año 2013 y en el 2014. Para la campaña del año 2015 se escogieron 50 campos. Dentro de estas campañas obtuvieron más de 120 candidatos jóvenes a supernova. Sin embargo, no se logró encontrar en estas rastros de SBO (figura 2.3) evento que comprende uno de los principales objetivos de HiTS (razón por la cual buscan desarrollar una herramienta que les permita la detección temprana de supernovas de tipo II).

2.3. El filtro de Kalman

La evolución determinística de un sistema físico en el tiempo es conocido sí el estado del sistema es medido con absoluta precisión en cada instante de tiempo (i.e. en un entorno donde es posible despreciar fenómenos cuánticos). Sin embargo toda medición está sujeta a incertezas finitas. Para sistemas los cuales son observados entre intérvalos prolongados de tiempo, se prevee que las diferen-

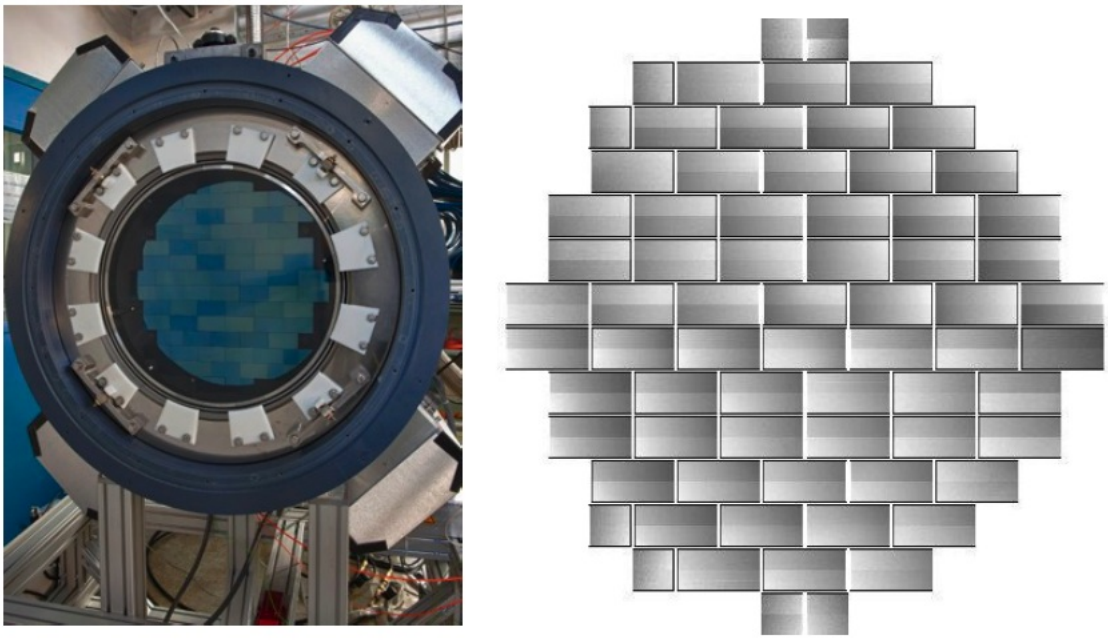


Figura 2.4: A la izquierda, estructura de la cámara DECam poblada con 62 chips CCDs. A la derecha, imagen *flat field* desde DECam.

cias entre los estados estimados y los medidos se incrementen con el tiempo. Para la obtención de predicciones lo más confiables posible se requiere que el sistema sea regularmente monitoreado y sus estados estimados futuros puedan ser considerados confiables en un lapso de tiempo en apropiado.

Los filtros de Kalman son métodos que proveen un trade-off entre los valores esperados del estado actual de un sistema y las mediciones que proporcionan información de su estado real. La aplicación de un filtro de Kalman está pensada como un proceso de dos fases:

1. **Fase predictiva:** Una estimación del estado actual del sistema que se basa en la estimación del estado previo o último estado. Esta predicción se denomina usualmente como estado estimado *a priori*.

2. **Fase correctiva:** La estimación del estado *a priori* es corregida con una medida actual para refinar la aproximación. Esta mejora se denomina *aproximación a posteriori*.

Típicamente estas fases de predicción y corrección se van alternando mientras se estudie el comportamiento físico de algún sistema.

2.3.1. Filtro de Kalman Básico

El filtro de Kalman Básico asume un comportamiento de sistema lineal y que las mediciones y las predicciones siguen una distribución Gaussiana.

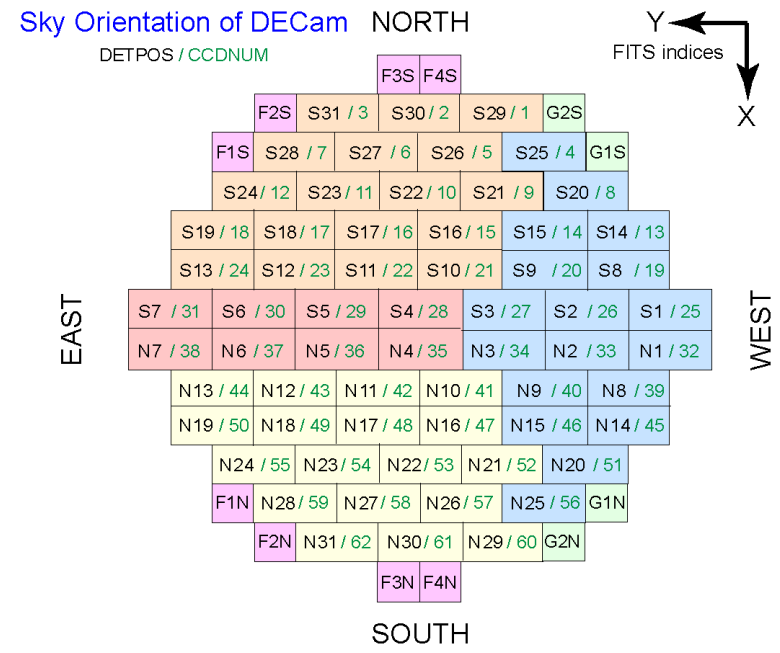


Figura 2.5: Orientaciones sobre el cielo y la huella espacial del arreglo de detectores en el plano focal. Se destacan los CCD cuya etiqueta comienzan con S o N, ya que estos corresponden a los detectores encargados de obtener las imágenes científicas. El etiquetado de estas componentes puede ser engañoso debido a que las iniciales de norte (North) y sur (South) están invertidas en relación a la orientación del cielo.

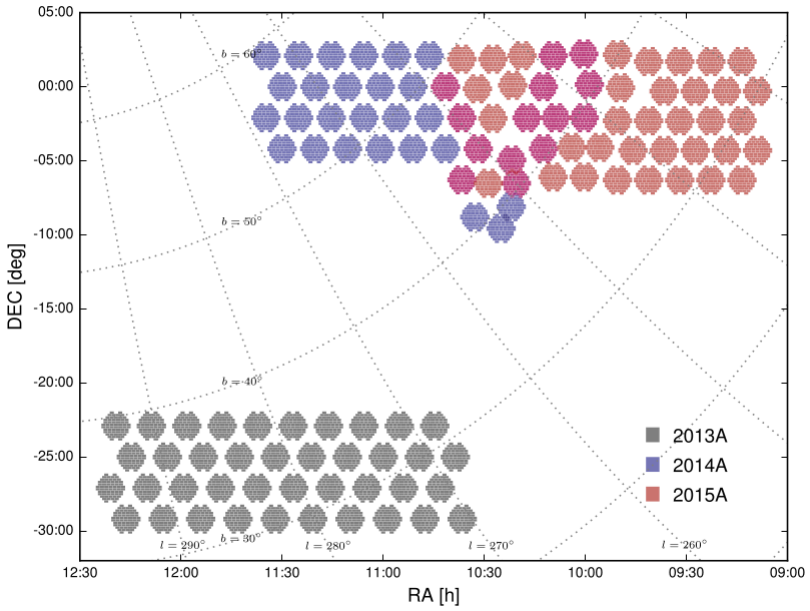


Figura 2.6: Distribución espacial de los campos observados durante los primeros semestres de los años 2013 (gris), 2014 (azul) y 2015 (naranjo). En tono rojo, los mismos campos del año 2015 y 2014 (superposición).

A continuación se describen las componentes más importantes en el desarrollo matemático del filtro:

- F_k**:** Matriz de transición de estado, de dimensiones $N \times N$ (N es el número de variables de estado).
- H_k**:** Matriz de transformación de estado a medición, $K \times N$ (K corresponde a las mediciones realizadas en un instante k de una variable de estado).
- Q_k**:** Matriz de covarianza del ruido del proceso ($N \times N$).
- R_k**:** Matriz de covarianza del ruido de las mediciones ($K \times K$).
- B_k**:** Matriz de control de entrada (contiene alteraciones que se querrían agregar al sistema de manera deliberada, por ejemplo, como la condición de parada de un vehículo en movimiento). Esta matriz es de dimensiones $N \times L$, donde L es la dimensión del vector de control de entrada u_k.

A continuación se hará uso de la notación en subíndices $m|n$, en las estimaciones de estado y covarianzas, para explicitar al instante de tiempo al cual pertenecen: m; y al instante de tiempo de donde se extrae la información: n.

En el instante $k-1$, se obtienen dos cantidades

- $\hat{x}_{k|k-1}$: Estado estimado a priori.
- $P_{k|k-1}$: Matriz de covarianza *a priori*.

Luego, en la fase de corrección se calculan:

- $\hat{x}_{k|k}$: Estado estimado *a posteriori*.
- $P_{k|k}$: Matriz de covarianza *a posteriori*.

Con estas variables, podemos describir las ecuaciones que explican la evolución del proceso de este algoritmo:

1. **Fase predictiva:**

   Estimación de estado y matriz de covarianza *a priori*.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \tag{2.1}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.2}$$

2. **Fase correctiva:**

   Estimación de estado y matriz de covarianza *a posteriori*.

$$\hat{z}_k = H_k \hat{x}_{k|k-1} \tag{2.3}$$

$$\tilde{z}_k = z_k - \hat{z}_k \tag{2.4}$$

La ecuación 2.4 describe la obtención de un residuo de la diferencia entre la predicción y la medida z_k. Posteriormente se calcula la matriz de covarianza entre residuos (2.5), con la que se calcula la ganancia de Kalman (ecuación 2.6).

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.5}$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.6}$$

Con la ganancia de Kalman calculada, se actualiza el valor de la estimación de estado (2.7) y matriz de covarianza a posteriori (2.8).

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{z}_k \tag{2.7}$$

$$P_{k|k} = (I_N - K_k H_k) P_{k|k-1} \tag{2.8}$$

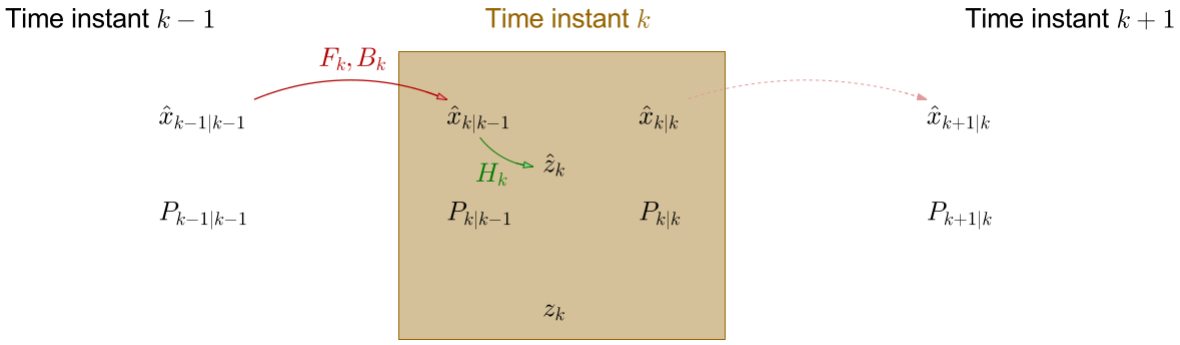


Figura 2.7: Representación del proceso de predicción (obtención de cantidades a priori) de las cantidades $\hat{x}_{k|k-1}$ y $P_{k|k-1}$; y de corrección (estimación a posteriori) para obtener las cantidades $\hat{x}_{k+1|k}$ y $P_{k+1|k}$.

2.3.2. Filtro de Kalman de Máxima Correntropía

El filtro de Kalman basado en correntropía máxima [1], difiere del filtro de Kalman tradicional (básico) en que no asume gaussianidad en las observaciones, considerando casos en que una señal puede ser perturbada por pulsos de ruido que sigan una distribución de cola pesada, y en esta oportunidad se utiliza el *criterio de correntropía máxima* para el proceso de correción.

Recordando que la correntropía es una medida de similitud entre dos variables aleatorias, supongamos, $X, Y \in \mathbb{R}$ con una distribución conjunta $F_{XY}(x, y)$, definimos la correntropía matemáticamente como:

$$V(X,Y) = E[\kappa(X,Y)] = \int \kappa(x,y) dF_{XY}(x,y) \tag{2.9}$$

Donde E representa al operador de esperanza y $\kappa(,)$ corresponde a un kernel Mercer invariante a desplazamientos (teorema de Mercer, [4]). Para este filtro se emplea una función de kernel Gaussiana, dado por

$$\kappa(x,y) = G_\sigma(e) = exp\left(-\frac{e^2}{2\sigma^2}\right) \tag{2.10}$$

Con este término definimos la *función de costo* de máxima correntropía.

$$J_{MCC} = \frac{1}{N}\sum_{i=1}^{N} G_\sigma(e(i)) \tag{2.11}$$

La ecuación 2.11 representa la función a máximizar. Su maximización calcula la estimación corregida para el estado en el instante k.

$$\hat{x}_k = argmax_x(J_{MCC}) = argmax_x(\sum_{i=1} G_\sigma(e_i(k))) \tag{2.12}$$

Para obtener el máximo de correntropía se procede a calcular el error residual $\tilde{e}_i$ (ecuación 2.13)

$$\tilde{e}_i = d_{i,k} - w_{i,k}\hat{x}_{t-1,k|k} \tag{2.13}$$

Con estos residuos definimos las matrices diagonales 2.14 y 2.15. La primera matriz corresponde a evaluaciones del kernel en errores estimados de predicción y la segunda en errores propios de las observaciones (ruido).

$$\tilde{C_{x,k}} = diag(G_\sigma(\tilde{e}_{1,k}), ..., G_\sigma(\tilde{e}_{n,k})) \tag{2.14}$$

$$\tilde{C_{y,k}} = diag(G_\sigma(\tilde{e}_{n+1,k}), ..., G_\sigma(\tilde{e}_{n+m,k})) \tag{2.15}$$

La siguiente expresión corresponde una transformación de la covarianza del ruido de las mediciones usando una descomposición de Choleski ($B_{r,k}$) en el instante k ([2]).

$$\tilde{R}_k = B_{r,k}\tilde{C}_{y,k}^{-1}B_{r,k|k-1}^T \tag{2.16}$$

Luego se calcula la transformación de la predicción de la matriz de covarianza de las estimaciones de estado, $P_{k|k-1}$ (ecuación 2.17).

$$\tilde{P}_{k|k-1} = B_{p,k|k-1}\tilde{C}_{x,k}^{-1}B_{p,k|k-1}^T \tag{2.17}$$

Posteriormente se calcula la ganancia de Kalman para este nuevo sistema.

$$\tilde{K} = \tilde{P}_{k|k-1}H_k^T(H_k\tilde{P}_{k|k-1}H_k^T + \tilde{R}_k)^{-1} \tag{2.18}$$

Finalmente la actualización de la esttimación de estado para el instante k queda como:

$$\hat{x}_{t,k|k} = \hat{x}_{k|k-1} + \tilde{K}_k(y_k - H_k\hat{x}_{k|k-1}) \tag{2.19}$$

Las ecuaciones de la 2.13 a la 2.19 se repiten secuencialmente hasta satisfacer la condición:

$$\frac{\parallel \hat{x}_{t,k|k} - \hat{x}_{t-1,k|k} \parallel}{\parallel \hat{x}_{t-1,k|k} \parallel} \leq \epsilon \tag{2.20}$$

El valor de ϵ es definido por el usuario y corresponde a un criterio de detención (el algoritmo puede detenerse definiendo un máximo en el número de pasos).

$$P_{k|k} = \left(I - \tilde{K}_k H_k\right) P_{k|k-1} \left(I - \tilde{K}_k H_k\right)^T + \tilde{K}_k R_k \tilde{K}_k^T \tag{2.21}$$

Finalmente se calcula la matriz de covarianza para el instante actual, k (expresión 2.21).

2.3.3. Filtro de Kalman Unscented

1. **Fase predictiva:**

   En esta versión del filtro [9] ya no se habla de matrices de transición de estado, F_k, ni de matrices de transformación de estado-a-medición, H_k, sino más bien de funciones diferenciables f y h respectivamente para describir la transición de estados y la transformación de estos a estimaciones a priori. Sin embargo, previo a estas transiciones se deben seleccionar 2N+1

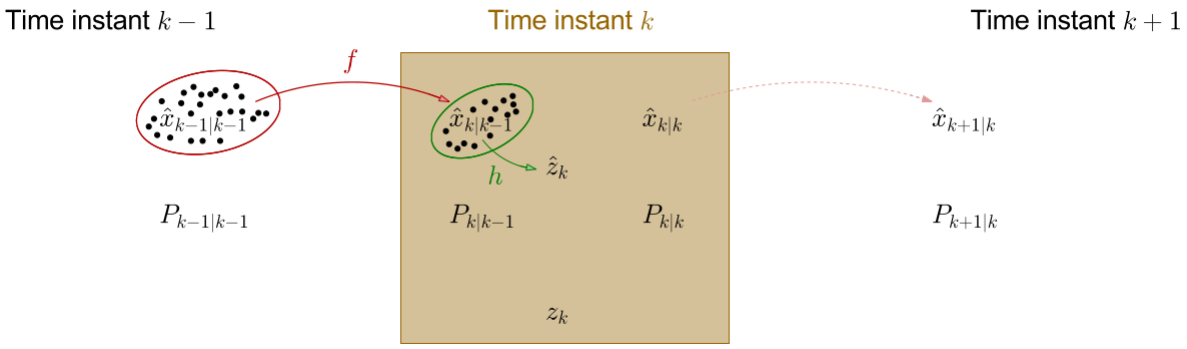


Figura 2.8: Representación del funcionamiento del filtro UKF. En esta oportunidad se hace uso de la función f y h para obtener las transformaciones $x_{k-1} \rightarrow x_k$ y $x_k \rightarrow z_k$. Esto se logra con la evaluación de de los 2N+1 puntos generados durante la etapa de predicción (y posteriormente en la etapa de corrección).

puntos representativos alrededor de $\hat{x}_{k-1|k-1}$ y evaluar estos en la función no lineal f, para obtener las estimaciones de $x_{k|\hat{k}-1}$ y $P_{k|k-1}$.

La generación de los 2N+1 puntos, se realiza a partir de la última estimación $\hat{x}_{k-1|k-1}$ de la siguiente forma

$$\begin{aligned} \bar{x}^0_{k-1|k-1} &= \hat{x}_{k-1|k-1} \\ \bar{x}^0_{k-1|k-1} &= \hat{x}_{k-1|k-1} + \chi_i, \quad \forall i \in [1, N] \\ \bar{x}^0_{k-1|k-1} &= \hat{x}_{k-1|k-1} - \chi_{i-N}, \quad \forall i \in [N+1, 2N] \end{aligned} \tag{2.22}$$

Donde la cantidad χ_i corresponde a la i-ésima columna de la *raíz cuadrada* de la matriz:

$$(N + \lambda)P_{k-1|k-1} \tag{2.23}$$

Esta matriz (ec. 2.23) puede obtenerse a partir de la descomposicón de Choleski. Por otro lado los puntos sigma se generan junto a dos conjuntos de pesos: $\{w^i_x\}$ y $\{w^i_p\}$. El primer conjunto se emplea en la estimación del estado y la predicción de la medida, mientras que el segundo conjunto es usado para obtener las matrices de covarianza relacionadas con el método. Estos pesos son definidos como:

$$\begin{aligned} w^0_x &= \frac{\lambda}{N + \lambda} \\ w^0_p &= w^0_x + 1 - \alpha^2 + \beta \\ w^i_x &= w^i_p = \frac{1}{2(N + \lambda)} \\ \sum_i^{2N} w^i_x &= 1 \end{aligned} \tag{2.24}$$

De las ecuaciones 2.24 se desprende que los pesos w_x^i son normalizados. Por otro lado, el parámetro λ se puede escribir en términos de los valores de $\alpha \in (0, 1]$ y *kappa* según la expresión 2.25

$$\lambda = \alpha^2(N + \kappa) - N \tag{2.25}$$

Los parámetros α, β y κ deben ser ajustados acorde al problema que se está estudiando.

Con esto, es posible escribir las ecuaciones de la fase predictiva.

- Estimación a priori de los estados

$$\hat{x}_{k|k-1} = \sum_{i=0}^{2N} w_x^i f(\bar{x}_{k-1|k-1}^i) \tag{2.26}$$

- Estimación a priori de la matriz de covarianza

$$P_{k|k-1} = \sum_{i=0}^{2N} w_p^i \left(f(\bar{x}_{k-1|k-1}^i) - \hat{x}_{k|k-1}\right)\left(f(\bar{x}_{k-1|k-1}^i) - \hat{x}_{k|k-1}\right)^T + Q_k \tag{2.27}$$

2. **Fase correctiva:**

Durante la fase de corrección, nuevamente se seleccionan 2N+1 puntos representativos, alrededor de $\hat{x}_{k|k-1}$. Estos posteriormente son evaluados en la función no-linear h.

$$\begin{aligned} \bar{y}_{k-1|k-1}^0 &= \hat{x}_{k|k-1} \\ \bar{y}_{k-1|k-1}^i &= \hat{x}_{k|k-1} + \psi_i, \quad \forall i \in [1, N] \\ \bar{y}_{k-1|k-1}^i &= \hat{x}_{k|k-1} - \psi_{i-N}, \quad \forall i \in [N+1, 2N] \end{aligned} \tag{2.28}$$

La cantidad ψ_i representa el i-ésima columna de la matriz de *raíz cuadrada* $(N + \lambda)P_{k|k-1}$.

Las ecuaciones del proceso de corrección, por tanto, quedan como sigue:

- Predicción de las medidas:

$$\hat{z}_k = \sum_{i=0}^{2N} w_x^i h(y_{k|k-1}^{-i}) \tag{2.29}$$

- Los residuos de las mediciones pueden obtenerse como:

$$\tilde{z} = z_k - \hat{z}_k \tag{2.30}$$

- La matriz de innovación:

$$S_k = \sum_{i=0}^{2N} w_p^i (h(y_{k|k-1}^{-i}) - \hat{z}_k)(h(y_{k|k-1}^{-i})^T + R_k \tag{2.31}$$

- La matriz de covarianza cruzada de estado a medida se describe:

$$C_k = \sum_{i=0}^{2N} w_p^i (f(\bar{x}_{k-1|k-1}^i) - \hat{x}_{k|k-1})(h(y_{k|k-1}^{-i}) - \hat{z}_k)^T \tag{2.32}$$

- La ganancia óptima finalmente queda:

$$K_k = C_k S_k^{-1} \tag{2.33}$$

- La estimación *a posteriori* de estado:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{z} \tag{2.34}$$

- Por otro lado, la ecuación para la matriz de covarianza:

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T \tag{2.35}$$

Los pesos w_x^i y w_p^i son los mismos calculados en la expresión 2.24, de la fase de predicción.

2.4. Laboratorio nacional de computación de alto desempeño (NLHPC)

El National Laboratory for High Performance Computing (NLHPC) es un proyecto asociativo, financiado por el PIA de CONICYT el cual dispone de un potente sistema computacional el cual está disponible a la comunidad científica y académica nacional (instituciones de investigación, industria y universidades), estimulando su uso en el desarrollo de áreas de investigación que requieran de herramientas computacionales robustas que deban ser usadas de manera intensiva.

Para la realización de esta tesis se hará uso de uno de los nodos del clúster de Leftraru, el supercomputador del NLHPC disponible para la comunidad investigadora, llegado el 2014 a las instalaciones del Centro de Modelamiento Matemático (CMM) de la Universidad de Chile.

- 132 nodos de cómputo HP (128 nodos HP SL230 y 4 nodos HP SL250), cada uno con dos procesadores de 10 cores Intel Xeon Ivy Bridge E5-2660 V2.
- 2640 núcleos

- 6.25 TB de RAM
- 274TB de almacenamiento Lustre (DDN EXAScaler)
- 12 co-procesadores Intel Xeon Phi5110p de 2 TFlops
- Capacidad de cómputo de 70 TFlops

3 | ¿Qué se entiende por *linealidad?*

De acuerdo a la sección del capítulo anterior, referente a las supernovas de tipo II2.1, se detalló el comportamiento de la luminosidad (o flujo) de este tipo de objetos. Esta curva de luz sin embargo no describe un comportamiento lineal2.2, por otro lado es posible linealmente *por trozos*, es decir, localmente.

En este trabajo, y en el trabajo de P. Huentelemu [8] se asume esta linealidad local, sin embargo esta asunción puede

3.1. El filtro de Kalman para un sistema localmente lineal

El filtro de Kalman tradicional o básico es derivado usando el MMSE (minimum mean square error) el cual funciona bien bajo condiciones de gaussianidad. Sin embargo cuando las señales son no-gaussianas o cuando el sistema estudiado es perturbado por impulsos de distribución de cola pesada el desempeño del filtro de Kalman básico se deteriora visiblemente, ya que el MMSE sólo captura el error hasta un segundo orden y es sensible a outliers [1].

Por este motivo se desarrolló en conjunto al filtro tradicional, un filtro de Kalman basado en correntropía máxima, que fuese insensible a outliers y robusto condiciones de no-gaussianidad.

4 | Desempeño del programa original

A continuación se exponen los resultados de diferentes pruebas realizadas sobre el programa original con la finalidad de medir su desempeño computacional en términos de tiempo de ejecución y uso de memoria.

Estas pruebas se llevaron a cabo en un computador personal de 8 GB (DDR4) de RAM. La razón de esta medida (que no se hayan ejecutado en Leftraru) tuvo que ver con que el código original contiene demasiadas líneas con el comando `glob`[1] de Python el cual lista reiteradamente el contenido de los directorios de los archivos lo que finalmente termina saturando el nodo destinado para el lanzamiento del programa (en Leftraru el programa no se ejecuta de la misma forma que lo haría localmente, ya que recorre la lista de 93 supernovas de HiTS, repitiendo para cada una de ellas el mismo proceso). Por esto último, los administradores del sistema del NLHPC sugirieron modificaciones al programa original, sin embargo, en pos de continuar con el espíritu de esta tesis, se optó por efectuar los experimentos de manera local (usando un computador personal) evitando así introducir más modificaciones (se adaptó el programa para Python 3.5, ya que originalmente estaba para 2.7 agregando cambios menores como la forma de imprimir mensajes en consola (`print`)).

Cabe destacar que el problema anteriormente descrito comenzó a presentarse una vez que el autor original del mismo pudo corregir la revisión de los nuevos candidatos en el mes de Junio del presente año, debido a que la primera versión del programa sólo estaba verificando la presencia de una supernova conocida y no estaba revisando los potenciales nuevos candidatos. Esto se agregó como una continuación del programa repitiendo pasos como el cálculo de flujos, calculo de matrices de estados y sus respectivas predicciones usando algún tipo de filtro de Kalman, etc. El diagrama de la figura 4.1 entrega una perspectiva general de la secuencia de pasos que realiza el programa.

[1] https://docs.python.org/3.5/library/glob.html

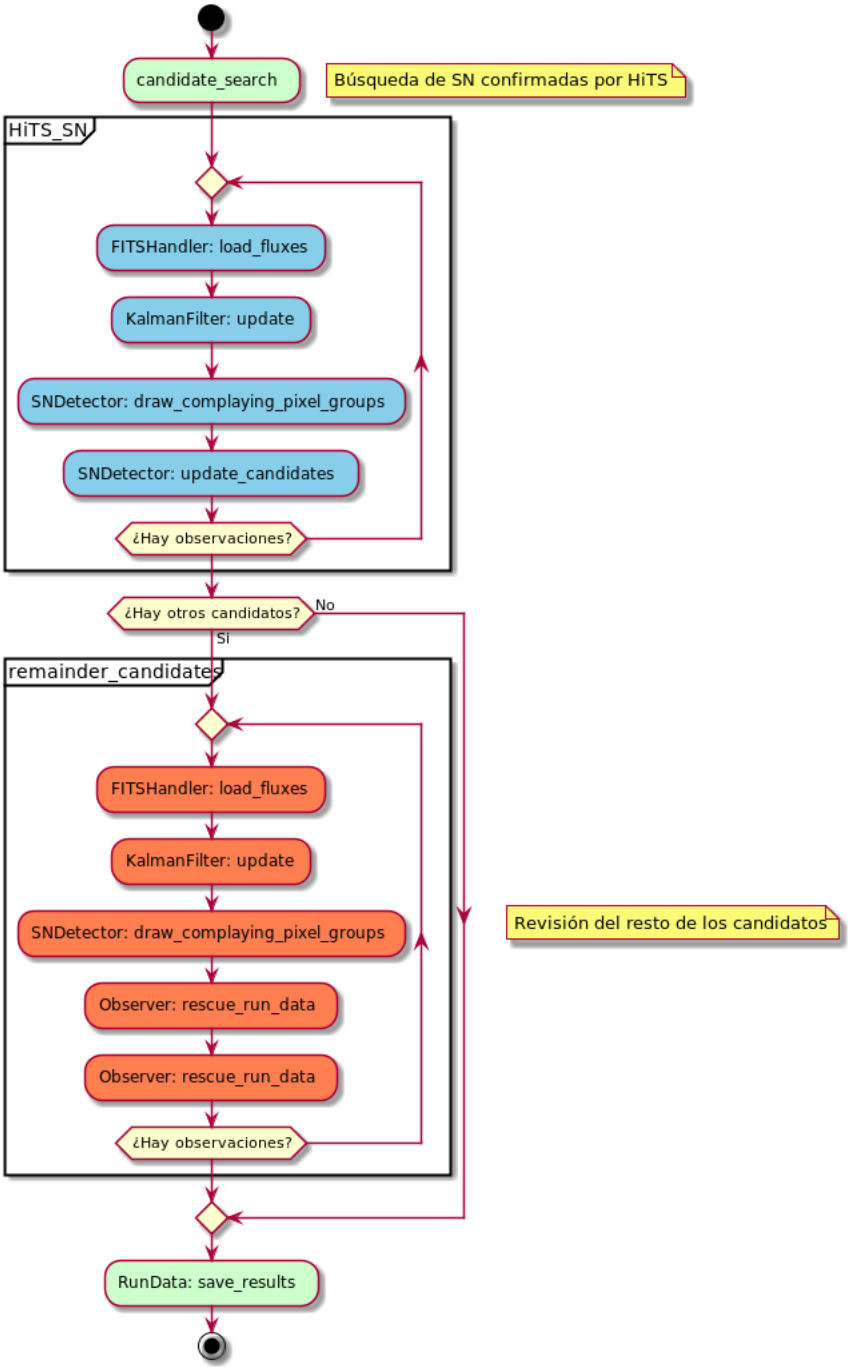


Figura 4.1: Diagrama de actividad del programa original. Se aprecían dos ciclos principales: el primero está destinado a la búsqueda de una supernova de HiTS, y el segundo a la revisión de la lista de posibles candidatos encontrados durante la verificación de la supernova de HiTS. Notar que hay pasos que se repiten en la realización de ambos análisis.

4.1. Tiempo de ejecución

El estudio del tiempo de ejecución del programa se realizó usando la función `getrusage` de la librería `resource` de Python, midiendo el tiempo de usuario en segundos. Las mediciones se rea-

lizaron sobre tres conjuntos de datos (las cuales contienen alguna supernova detectada por HiTS) seleccionados al azar: SN14, SN18 y SN80. En cada uno de ellos comprende secuencias de 26 observaciones.

ID	**Cálc. Flujos** [s]	**Aplic. KF** [s]	**Agrup. Pixeles** [s]	**Actual. Candidatos** [s]
SN14	320.98	29.39	39.98	0.01
SN18	290.48	23.89	36.64	0.01
SN80	297.79	25.40	36.88	0.01
Media	303.08	26.23	37.83	0.01
$\bar{t}/Obs$	11.66	1.01	12.61	0.00

Cuadro 4.1: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman Básico.

ID	**Cálc. Flujos** [s]	**Aplic. KF** [s]	**Agrup. Pixeles** [s]	**Guardar resultados** [s]
SN14	333.59	36.13	42.22	0.09
SN18	289.45	24.15	36.87	0.05
SN80	297.90	26.39	37.59	0.06
Media	306.98	28.89	38.89	0.07
$\bar{t}/Obs$	11.81	1.11	1.50	0.00

Cuadro 4.2: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman Básico.

ID	**Cálc. Flujos** [s]	**Aplic. KF** [s]	**Agrup. Pixeles** [s]	**Actual. Candidatos** [s]
SN14	327.92	731.98	38.42	0.01
SN18	290.05	554.62	35.00	0.00
SN80	309.98	628.83	37.80	0.01
Media	309.32	638.48	37.07	0.01
$\bar{t}/Obs.$	11.90	24.56	1.43	0.00

Cuadro 4.3: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman de Máxima Correntropía.

4.2. Uso de memoria

La memoria ocupada por el programa se midió en términos de MiB (Mebibyte) usando la librería `memory_profiler`. Posteriormente las mediciones en la unidad previamente mencionada fueron pasadas a MB[2]

[2] $1MiB \simeq 1.049$

ID	Cálc. Flujos [s]	Aplic. KF [s]	Agrup. Pixeles [s]	Guardar resultados [s]
SN14	302.28	631.17	36.98	0.04
SN18	311.63	660.51	35.68	0.06
SN80	306.22	610.60	36.37	0.04
Media	306.71	634.09	36.34	0.05
$\bar{t}/Obs.$	11.80	24.39	1.40	0.00

Cuadro 4.4: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el periódo de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman de Máxima Correntropía.

ID	Búsqueda SN	Revisión candidatos	Tiempo total
SN14	390.36	412.03	802.39
SN18	351.02	350.52	701.54
SN80	360.08	361.94	722.02
Media	367.15	374.83	741.98
$\bar{t}/Obs.$	14.12	14.42	28.54

Cuadro 4.5: Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman Básico.

ID	Búsqueda SN	Revisión candidatos	Tiempo total
SN14	1098.33	970.47	2068.80
SN18	879.67	1007.88	1887.55
SN80	976.62	953.23	1929.85
Media	367.15	374.83	741.98
$\bar{t}/Obs.$	14.12	14.42	28.54

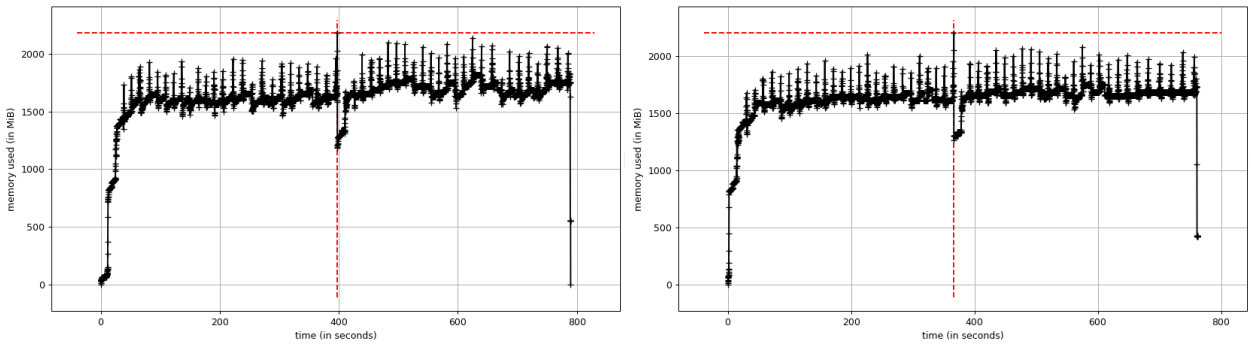
Cuadro 4.6: Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman de Máxima correntropía.

ID	Memoria [MB]
SN14	2290.10
SN18	2314.17
SN80	2191.82

Cuadro 4.7: Memoria principal (en unidades de MB) usada durante la ejecución del programa original usando la versión básica del filtro de Kalman.

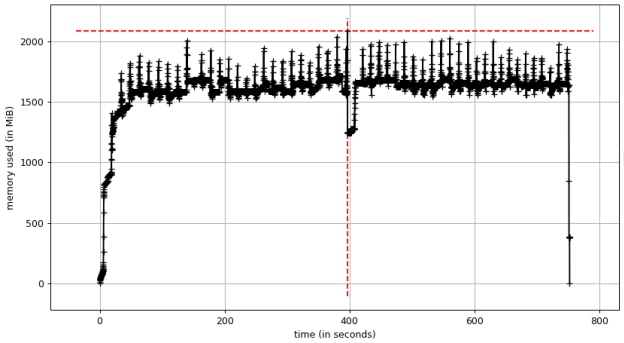
ID	Memoria [MB]
SN14	3320.12
SN18	3248.78
SN80	3320.12

Cuadro 4.8: Memoria principal (en unidades de MB) usada durante la ejecución del programa original usando filtro de Kalman de máxima correntropía.



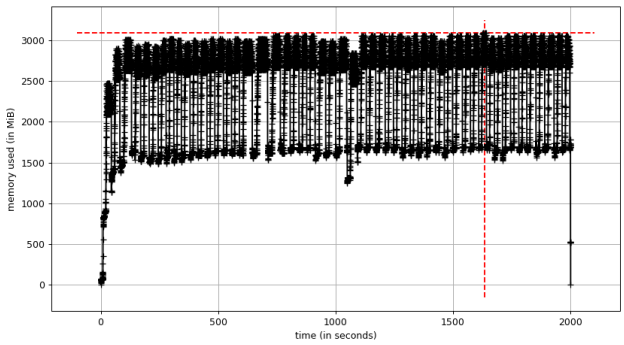
(a) *Memoria ocupada en SN14*

(b) *Memoria ocupada en SN18*

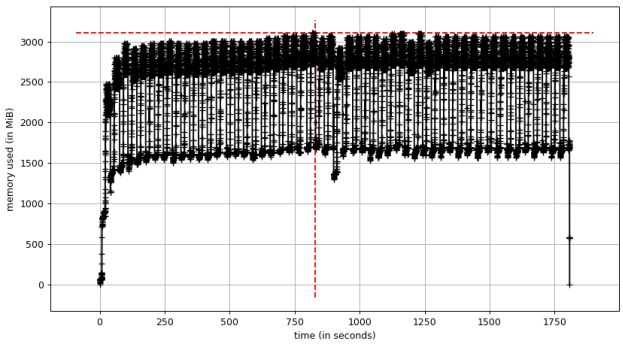


(c) *Memoria ocupada en SN80*

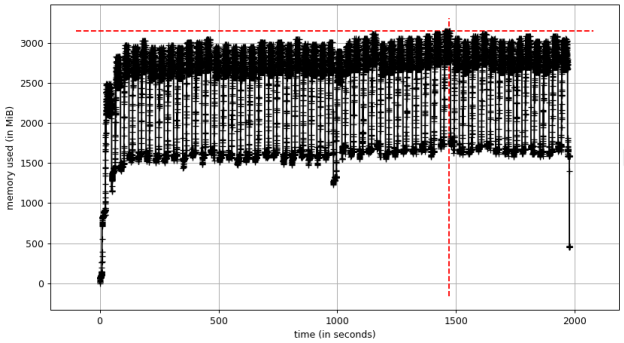
Figura 4.2: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman Básico.



(a) *Memoria ocupada en SN14*



(b) *Memoria ocupada en SN18*



(c) *Memoria ocupada en SN80*

Figura 4.3: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman de correntropía máxima.

5 | Refactoring

El refactoring del código original se realizó en Python en su versión 3.5 (versión diferente a la que el programa fue implementado inicialmente: 2.7). Además se mantuvieron casi la totalidad de las librerías cambiando únicamente Pymorph[1] por Mahotas[2] (ambas librerías desarrolladas por el mismo autor, Luis P. Coelho[3]), debido a que la primera dejó de ser mantenida desde el 2010 y contempla adaptación para Python 3.

5.1. Manejo de datos de entrada

Toda la información de entrada es manipulada y procesada en la clase DataPicker. Esta clase se inicializa recibiendo un path hacia un archivo de configuración (ver apéndice,B .2) que contiene tanto las rutas a los archivos así como los nombres de estos en términos de expresiones regulares, semestre a los que corresponde la secuencia de observaciones (los dos últimos dígitos del año concatenados con la letra A en caso de corresponder al primer semestre o B al segundo), el campo (representado como un número de dos dígitos, comenzando con cero para valores menores a 10) y el detector CCD (cadena de tres carácteres donde el primero de ellos describe a que grupo de detectores corresponde: N o S (ver figura 2.5); además de un número entero que va de 1 a 36) como strings.

- `maskDir`: Directorio donde se almacenan las imágenes máscara (imágenes que identifican píxeles que no deben ser considerados).
- `scienceDir`: Directorio donde se almacenan las imágenes científicas (imágenes base ya preprocesadas).
- `diffDir`: Directorio donde se almacenan las imágenes de diferencia (resta entre las imágenes base y su científica).

[1] https://pythonhosted.org/pymorph/
[2] https://mahotas.readthedocs.io/en/latest/
[3] https://github.com/luispedro

- `psfDir`: Directorio donde se encuentran los modelos de psf1 usados para la determinación del flujo.
- `invDir`: Directorio que guarda las imágenes correspondientes a la varianza inversa (*peso* de cada pixel en términos de ruido: a menor peso, mayor ruido).
- `afluxDir`: Directorio que contiene los archivos de extensión `NPY` dentro de los cuales se guarda el valor de la variable `aflux`.
- `maskRegEx`: Expresión regular con la que es posible identificar el nombre de las imágenes máscara en disco siguiendo el path `maskDir`.
- `scienceRegEx`: Expresión regular con la que es posible identificar el el nombre de las imágenes científicas en disco siguiendo el path `scienceDir`.
- `diffRegEx`: Expresión regular con la que se identifican el nombre de las imágenes de diferencia en disco siguiendo el path `diffDir`.
- `invRegEx`: Expresión regular con la que es posible identificar el nombre de las imágenes de la varianza inversa siguiendo el path `invDir`.
- `afluxRegEx`: Expresión regular con la que se identifica el nombre de los archivos *match* que contienen el valor de `aflux` contenidos en el path `afluxDir`.
- `psfRegEx`: Expresión regular que describe el nombre de las imágenes que guardan el modelo de PSF en el directorio `psfDir`.
- `imgHeight`: Valor entero que indica alto de la imagen.
- `imgWidth`: Valor entero que indica ancho de la imagen.
- `only_HiTS_SN`: Es `yes` en caso de querer encontrar únicamente las supernovas conocidas, y `no` en caso de desear analizar la existencia de otros candidatos posibles (aparte de las supernovas de HiTS).
- `results`: Directorio donde se irán almacenando los resultados tanto de las observaciones como de las predicciones realizadas con los filtros de Kalman de los potenciales candidatos a supernova.

DataPicker maneja la lectura y preparación de las imágenes a ser analizadas, mientras que un segundo proceso correspondiente a la lectura de resultados anteriores (guardados en un archivo de texto plano) se describirá en la próxima sección de nuevas funcionalidades 6.

5.1.1. Lectura y preparación de imágenes

A continuación se enumeran los diferentes métodos que intervienen en la recolección de los datos a ser leídos:

- `config_reg_expressions(semester, field, ccd)`

  Este método recibe como parámetros el semestre (`semester`), el campo (`field`) y el ccd que la misma clase recibe de entrada. Con estos strings se establecen las rutas de los directorios de las imágenes y las expresiones regulares de los nombres de las mismas.

- `collect_data()`

  Esta función se encarga de recolectar la ruta completa de las diferentes imágenes (máscarac, científicas, de diferencia, etc.). Para esta finalidad se hace uso del método `walking_through_files`.

- `walking_through_files(regex, dir)`

  Método con el cual se recorren las rutas definidas en los pasos anteriores y se agrupan los nombres completos (directorio incluido) de las imágenes ubicadas en el directorio `dir` y posean un nombre de patrón que siga la expresión regular `regex`.

- `filter_science_images()`

  Filtra imágenes científicas de acuerdo a su airmass 2, seleccionando las fechas en que las observaciones fueron hechas en términos de *día juliano modificado* o *Modified Julian Date* (MJD) como variables de punto flotante.

- `select_fits(dir)`

  Selecciona y ordena los elementos de la lista de imágenes de formato fits del directorio `dir` en orden cronológico (i.e. de acuerdo a la secuencia de MJD encontrada en `filter_science_images()`).

- `select_npys(dir, ref_dir, init_index, n_pos, rest_len)`:

  Selecciona los archivos de extensión NPY que se encuentran en el directorio `dir`. Los nombres son filtrados en relación a las imágenes listadas de `ref_dir` según el paso de `select_fits(dir)` (es decir, debe ser un directorio que contenga imágenes FITS). `init_index`, `n_pos` y `rest_len` son enteros usados para extraer substrings específicos de los nombres de los archivos.

5.2. Determinación de flujos

El proceso de la obtención del flujo se simplificó, eliminando la clase FitsHandler del programa original. Debido a la posibilidad de hacer los métodos de esta clase estáticos se implementó un script Python denominada `utils` para contener estas rutinas e implementarlas estáticamente.

Los métodos que participan en la rutina de calculo de flujo son:

- `naylor_photometry(invvar, diff, psf)` [7]:
  Calcula el producto del flujo por su variaza. Retorna el producto y la varianza. Para esto obtiene el flujo a partir de la imagen PSF entregada (`psf`) y del producto entre la imagen diferencia y la de varianza inverza (`diff` y `invvar` respectivamente).

- `calc_fluxes(diff, psf, invvar, aflux):` Calcula el flujo y su varianza gestionando la entrada y la salida de `naylor_photometry(invvar, diff, psf)`. Los valores NaN son transformados a valores de punto flotante de constante 0.001.

5.3. Filtros originales

La refactorización de los filtros de Kalman originales implicó la implementación de nuevas clases e interfaces para el desarrollo del patrón propuesto: Strategy. A continuación se presentan cada una de ellas:

- **IPredict:** Interface que describe el comportamiento de la función PREDICT de un filtro de Kalman. `predict` recibe como parámetros el paso de tiempo (Δt), la matriz de estados, la matriz de covarianza de estados, y las predicciones de las matrices de estado y covariaza determinadas (con la finalidad de reciclar estas variables) en el paso anterior. Su firma queda como:

  ```
  predict(delta_t, state, state_cov, pred_state, pred_cov)
  ```

  Este filtro entrega finalmente las matrices de estado y covarianza de estado predichas.

- **ICorrect:** Interface que describe el comportamiento de la función CORRECT de un filtro de Kalman. `correct` recibe como parámetros el matriz de flujo (`z`) y de varianza de las observaciones (`R`), la matriz de estados predicha, la matriz de covarianza predicha, la matriz de estado y la matriz de covarianza (para sobreescritura) obtenidas en el paso anterior. Su firma queda como:

```
correct(z, R, pred_state, pred_cov, state, state_cov)
```

Este filtro entrega finalmente las matrices de estado y covarianza de estado corregidos.

5.3.1. Strategies de predicción

LinearPredict: Clase que extiende de IPredict. Implementa método `predict` que será usado tanto por el filtro básico como el de máxima correntropía. Su instanciación recibe como argumento `sigma_a` (desviación estándar del modelo, asumiendo una distribuión gaussiana en la distribución de las observaciones).

5.3.2. Strategies de corrección

BasicCorrect: Clase que extiende de ICorrect. Implementa método `correct` que será usado para el tipo de filtro de Kalman Básico.

MCCorrect: Clase que extiende de ICorrect. Implementa método `correct` que será usado para el tipo de filtro de Kalman de máxima correntropía.

5.3.3. Filtros refactorizados

- **KalmanFilter:** Clase abstracta padre de los subtipos BasicKalmanFilter y MCKalmanFilter. Posee los métodos abstractos `predict` y `correct`, que son definidos de acuerdo a las estrategias de predicción y correción descritas previamente.
- **BasicKalmanFilter:** Representa el filtro básico de Kalman. Está compuesto por las estrategias `LinearPredict` y `BasicCorrect`.
- **MCKalmanFilter:** Representa el filtro de máxima correntropía. Está compuesto por las estrategias `LinearPredict` y `MCCorrect`.

5.4. Detección de candidatos

La detección de candidatos en el programa original se realiza en SNDetector sin embargo, durante este refactoring se descompuso el proceso de reconcimiento de fuentes estelares (grupo de pixeles brillantes) del proceso de selección de candidatos, siendo este la continuación del primero. Por tanto se crearon las clases SourceFinder para el filtrado y agrupamiento de pixeles y TP-Detector (transient phenomena detector) para la revisión de las *fuentes* encontradas en el paso anterior.

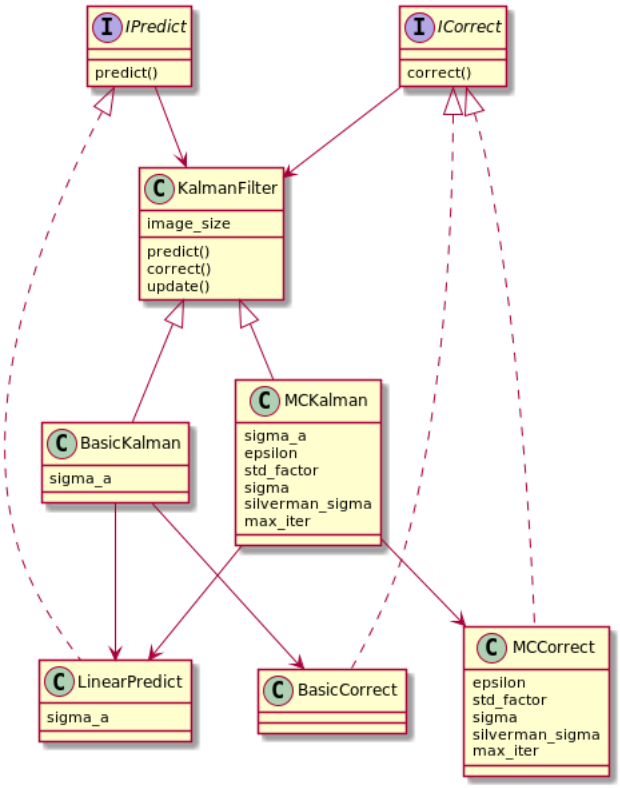


Figura 5.1: Familia de filtros de Kalman y patrón strategy usado en la implementación de los métodos `predict` y `correct`.

5.4.1. SourceFinder

La clase SourceFinder posee los siguientes métodos:

- `pixel_discard`:

  Método en el que se realiza el descarte de pixeles de forma individual, siguiendo los siguientes criterios:

  1. Si el estado (flujo) calculado para un pixel es menor que el umbral dado.
  2. Si la tasa de cambio estado (cambio o variación de flujo) es menor que el umbral de variación de flujo multiplicado por el nivel de saturación alcanzado en esta variación (se define una tasa de saturación *aceptable*).
  3. Si un pixel de la imagen científica es menor a la mediana más un valor arbitrario (en este trabajo, siguiendo la línea de desarrollo de P. Huentelemu, se consideró 5.0) es descartado.
  4. Si las varianzas de flujo son mayores a 150.0 (valor arbitrario)
  5. Si las varianzas de la tasa de cambio de flujo (o velocidad de flujo) es mayor o igual a 150.0

6. Si los pixeles no caen en etiquetas de invalidación dentro de la máscara que ha sido procesada para marcar también los pixeles vecinos a los realmente defectuosos.
7. Si los pixeles no han caido dentro del descarte por superar la mediana estiamda a partir de cuatro épocas.

- `grouping_pixels(pixel_flags, mjd_index)`:
  Este método recibe como entradas las etiquetas determinadas en el paso anterior de validación de pixeles (y el descarte de aquellos que no cumplan con los siete criterios) en un arreglo de matrices denominada `pixel_flags`. Además recibe el índice de MJD (o fecha en día juliano modificado) correspondiente a la observación de tal fecha. La agrupación de pixeles se realiza gracias a funciones brindadas por la librería Mahotas, usando el método `label` para encontrar dominios cerrados en el mapa de pixeles validados.

- `filter_groups(science, flux, var_flux, state, base_mask, median_reject)`:
  Este método recibe la imagen científica, el flujo y su varianza, el estado determinado por el filtro de Kalman, la máscara, y un arreglo en el que se guardan los valores de las medianas por cada cuatro observaciones. El filtrado de grupos de pixeles se lleva a cabo bajo las siguientes reglas de descarte:

  1.
  2. Si no hay máximos locales dentro del grupo de pixeles encontrados dentro de la imagen científica.
  3. Si no hay máximos locales dentro del grupo de pixeles encontrados dentro de la matriz de flujo.
  4. Si no hay máximos locales dentro del grupo de pixeles encontrados dentro de la matriz velocidad de flujo.
  5. Si los valores de los pixeles superan la mediana local en imagen científica.
  6. Si el grupo posee algún pixel que doble el valor del flujo o de la imagen científica.
  7. Si el centro del grupo se encuentra etiquetado como defectuoso dentro de la máscara.
  8. Si el pixel del centro del grupo se encuentra rechazado al ser superior a la mediana de los pixeles de cuatro observaciones consecutivas.
  9. Si la varianza del flujo del pixel del centro del grupo es mayor al determinado por el umbral.

- `draw_complying_pixel_groups`:

  Este método es el que llama a `pixel_discard` para etiquetar pixeles para el descarte y no ser considerados posteriormente en el paso de agrupamiento al llamar a `grouping_pixels`. Luego se llama al método `filter_groups` para hacer el descarte a nivel grupal y obtener candidatos. La última llamada es para el método `save_data` que se encuentra en clase DataContent.

6 | Nueva funcionalidad

Se detallan en las siguientes secciones la nueva funcionalidad agregada (no incluye el manejo de los datos en DataPicker ya que se describe como parte del refactoring), con la que se pretende implementar la funcionalidad en línea de este programa.

6.1. Manejo de la rutina: RoutineHandler

La **rutina** se entiende como la ejecución completa del programa, con la cual se procesan todas las observaciones de acuerdo a una lista de CCDs y campos. Para esta finalidad se creó una clase llamada RoutineHandler la cual maneja los archivos de entrada de:

- Lista de campos, CCD y semestre en un archivo CSV (en este mismo orden) y con el encabezado `Field, CCD, Semester`.
- Diccionario de directorios y expresiones regulares de las ubicaciones de los archivos y sus nombres, respectivamente (archivo TXT).
- Diccionario de umbrales y valores relevantes en la ejecución del programa, así como el tipo de filtro a usar (archivo TXT).

Esta clase contiene los siguientes métodos:

- `process_settings()`:
  En este método se lee el archivo de diccionario de umbrales y valores con los que se configurará la ejecución completa del programa.

- `retrieve_kalman_filter(kalman_string)`:
  Corresponde a un método auxiliar que es invocado desde `process_settings` con el que se crea una instancia del filtro de Kalman a partir de la lectura del archivo de valores, de acuerdo al valor defindo por el usuario. Los tres strings válidos para la construcción de una instancia son: 'Basic', 'MCC' y 'UKF'. Si se entrega otro tipo de string, se levanta un error.

- `iterate_over_sequences()`:

  Recorre la lista de campos, CCDs y semestres entregada al programa con la consiguiente llamada a `routine`

- `routine(semester, field, ccd, results_path, last_mjd)`:

  Corresponde a la rutina que comprende el análisis de las observaciones de un semestre, campo y CCD específico.

6.2. Detección de fenómenos trasientes: TPDetector

Es una pequeña clase con la que se apoya el proceso de reconocimiento o detección de algún fenómeno trasiente en el comportamiento de la intensidad de los pixeles calculados después la ejecución de `routine` (es decir, una vez que se han obtenido los resultados con `routine` de RoutineHandler).

Esta clase posee los siguientes métodos:

- `look_candidates(results_path, field, ccd, semester)`:

  Con este método se agrupan los resultados obtenidos por campo, CCD y semestre en la ruta de los resultados (`results_path`), y carga los arreglos de los candidatos encontrados (de acuerdo a su coordenada central) y cuenta las veces que aparece cada uno en los archivos (un archivo por época u observación) con la finalidad de registrar *las veces que ha sido candidato.*

- `list_candidates(cand_mid_coords)`

  Con finalidades exploratorias, este método registra los candidatos (por pares de pixeles que describen el centro de los grupos encontrados) sin repetición, independiente de las veces que han aparecido.

6.3. DataContent

DataContent es una clase auxiliar que es usada principalmente para encapsular los resultados obtenidos durante el proceso de detección, tanto de los pixeles de los grupos encontrados por época, la lista de pixeles centrales de estos grupos, las matrices de etiquetas (para realizar seguimiento de las causas del descarte de los pixeles o grupo de pixeles), así como también los estados encontrados

por el filtro usado. Para guardar estos resultados se utiliza la función `savez` de la librería NUMPY, los archivos guardados son registrados en un archivo de extensión NPZ.

Para el guardado de los datos se implementó la función `save_results` que recibe como parámetros la ruta donde se quiere almacenar los resultados, el campo de observación, el CCD, el MJD y la matriz de estados junto con la covarianza de estados. El resto de los arreglos son definidos en las funciones `set_mid_coords` y `group_info`.

7 Resultados

7.1. Desempeño

7.1.1. Filtros básico y de máxima correntropía

7.1.2. Filtro unscented

7.2. Pruebas en Leftraru

7.2.1. Filtro básico

7.2.2. Filtro de máxima correntropía

7.2.3. Filtro unscented

8 | Conclusiones

Capítulo Adicional que no es apéndice

Apéndices

A . Glosario

1. **PSF**

   Corresponde a la respuesta instrumental a una fuente de luz puntual, cuya radiación debe atravesar la atmósfera terrestre y los lentes del telescopio. La distorsión puede ser interpretada como la convolución de la imagen por un kernel. [8]. Ver imagen 8.1

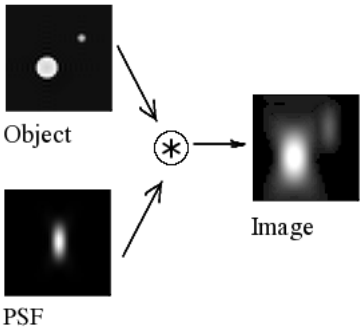


Figura 8.1: Ejemplo de distorsión de una fuente al aplicar un kernel de PSF específico. El resultado se observa en el cuado *Image*.

2. **Airmass**

   Es el largo del camino de que le toma a los rayos de una cuerpo celeste atravesar la atmósfera. A medida que los rayos van penetrando la atmósfera estos se van atenuando por la absorción y el proceso conocido como scattering.

3. **Fotometría de Naylor**

B . Refactoring

B .1. Librerías usadas para el refactoring

Versión de Python: 3.5

- `pandas`: **0.24.4**
- `matplotlib`: **2.2.2**
- `numpy`: **1.13.3**
- `mahotas`: **1.4.4**
- `astropy`: **3.0.2**

B .2. Archivo de entrada: configuración de paths

```
maskDir = /home/paloma/Documents/Memoria/data/Blind%s_%s/%s
scienceDir = /home/paloma/Documents/Memoria/data/Blind%s_%s/%s
diffDir = /home/paloma/Documents/Memoria/data/Blind%s_%s/%s
psfDir = /home/paloma/Documents/Memoria/data/Blind%s_%s/%s/CALIBRATIONS
invDir = /home/paloma/Documents/Memoria/data/Blind%s_%s/%s
afluxDir = /home/paloma/Documents/Memoria/data/Blind%s_%s/%s/CALIBRATIONS
maskRegEx = Blind%s_%s_%s_[0-9][0-9]_dqmask.fits.fz
scienceRegEx = Blind%s_%s_%s_[0-9][0-9]_image_crblaster_grid02_lanczos2.fits
diffRegEx = Diff_Blind%s_%s_%s_[0-9][0-9]t*-0[0-9]t*_grid02_lanczos2.fits
invRegEx = invVAR_Blind%s_%s_%s_[0-9][0-9]t*-0[0-9]t*_grid02_lanczos2.fits
afluxRegEx = match_Blind%s_%s_%s_[0-9][0-9]-0[0-9].npy
psfRegEx = psf_Blind%s_%s_%s_[0-9][0-9]t*-0[0-9]t*_grid0[1-2]_lanczos2.npy
```

C . Nueva funcionalidad

C .1. Modelo de archivo de almacenamiento de resultados

D . Unit-tests

D .1. Refactoring

- `test_input`

  Comprueba que outputs generados de la lectura de los datos de la nueva versión del programa sea identica a los generados por el código original. Esto se logra siguiendo el proceso de filtrado de imágenes científicas por airmass con la consiguiente secuenciación cronológica (en términos de MJD) del resto de las imágenes.

- `test_flux`

  Prueba el correcto funcionamiento de `calc_flux` mediante comparación de outputs de versión original y nueva.

- `test_basicKF`

  Comprueba que las salidas de la versión bśica del filtro de Kalman del programa refactorizado sea idéntico al de la versión original, usando secuencia de imágenes conocida.

- `test_MCKF`

  Comprueba que las salidas de la versión de máxima correntroía del filtro de Kalman del programa refactorizado sea idéntico al de la versión original, usando secuencia de imágenes conocida.

Referencias

[1] B. Chen, X. Liu, H. Zhao, and J. C. Príncipe. Maximum Correntropy Kalman Filter. *ArXiv e-prints*, Sept. 2015.

[2] B. Chen, X. Liu, H. Zhao, and J. C. Principe. Maximum correntropy kalman filter. *Automatica*, 76:70 – 77, 2017.

[3] F. Forster, J. C. Maureira, S. Gonzalez-Gaitan, G. Medina, G. Pignata, L. Galbany, J. San Martin, M. Hamuy, P. Estevez, R. C. Smith, K. Vivas, S. Flores, P. Huijse, G. Cabrera, J. Anderson, F. Bufano, T. de Jaeger, J. Martinez, R. Munoz, E. Vera, and C. Perez. HiTS real-time supernova detections. *The Astronomer's Telegram*, 7146, Feb. 2015.

[4] B. J Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. Lond. A*, 209(441-458):415–446, 1909.

[5] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[6] A. P. A. LSST Science Collaboration, A. J., and A. S. F. et al. *LSST Science Book, Version 2.0*. arXiv:0912.0201, 2009.

[7] T. Naylor. An optimal extraction algorithm for imaging photometry. *Monthly Notices of the Royal Astronomical Society*, 296(2):339–346, 1998.

[8] H. P. Filtro de Correntropía para detección de supernovas. *Tesis de magíster en ciencias de la ingeniería, mención eléctrica de la Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas*, 2016.

[9] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.