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

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

PALOMA PÉREZ GARCÍA

PROFESOR GUÍA:
SR. PABLO ESTEVEZ VALENCIA

PROFESOR CO-GUÍA:
SR. BENJAMÍN BUSTOS

MIEMBROS DE LA COMISIÓN:
SR. AIDAN HOGAN

SANTIAGO DE CHILE
MARZO 2019

RESUMEN DE LA MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN
POR: PALOMA CECILIA PÉREZ GARCÍA
FECHA: 2019
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA
PROF. CO-GUÍA: BENJAMÍNA BUSTOS

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

En este trabajo de título se describe un proceso de refactoring a una pipeline dedicada a la detección de fenómenos astronómicos transitorios como las supernovas que corresponden a eventos caracterizados por un incremento y un consecuente decremento de su luminosidad. Esta pipeline sobre la que se trabajó usa como método de detección una familia de filtros de estimación conocida como Filtros de Kalman, en particular sus versiones básica y de correntropía máxima.

El refactoring llevado a cabo apuntó a optimizar el manejo de los datos de entrada (tanto parámetros de funcionamiento como información a ser procesada, en este caso, imágenes) y a modularizar los métodos de estimación de los filtros de Kalman usando el patrón de diseño *Strategy* definido por sus procesos de predicción y corrección. Además se implementaron nuevas formas de guardar resultados y visualizar resultados conservando aspectos de las gráficas generadas por el software original.

Por otro lado, se propone un nuevo miembro para la familia de filtros de Kalman con el cual es posible usar una aproximación no-lineal para la detección de fenómenos transitorios de regimen creciente, conocido como filtro de Kalman Unscented.

...

Agradecimientos

Paloma Pérez García

Índice general

Índice de tablas

Índice de figuras

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) [10], el Visible and Infrarred Survey Telescope (VISTA) [7], el VLT Survey Telescope (VST) [2], el Dark Energy Camera Legacy Survey (DECaLS) [4] y el Hyper Suprime-Cam Subaru Strategic Program (HSC SSP) [1]. Estos surveys están caracterizados por un amplio *étendue* que consiste en el 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) [12] (que entrará en funcionamiento a mediados del 2022) continuarán revolucionando la era del big data en astronomía con étendues y cámaras CCD mucho más grandes de lo que se utiliza hasta el día de hoy. En particular se espera que el LSST produzca un número de alertas de fenómenos transitorios (avisos de objetos que cambian en el tiempo o espacio) del orden 10 millones 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.

Al día de hoy se han elaborado sondeos como el High Cadence Transient Survey (HiTS) [8] cuya finalidad ha sido la búsqueda de fenómenos transitorios 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 este trabajo se propone un método de detección de potenciales candidatos a supernova a través de la discriminación de píxeles que involucren un incremento en la 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 desarrollados para HiTS corresponden a miembros de una familia de filtros conocidos como *filtros de Kalman*.

El trabajo desarrollado por Pablo Huentelemu [14] (M.Sc.), propone el uso de los filtros clásico (o básico) [11] y de máxima correntropía [6] en el reconocimiento de supernovas jóvenes (o en su fase de crecimiento de luminosidad) de tipo II, por lo que se plantea la posibilidad de diseñar algún otro criterio de filtrado y de estudiar la variación de los resultados.

1.2. Objetivos

1.3. Objetivo general

El objetivo de esta tesis comprende la reestructuración y extensión de un programa existente destinado al análisis fotométrico de datos astronómicos con él cual se busca implementar un modelo de sistema de alertas que dé aviso del estado temprano de un fenómeno astronómico transitorio.

Paralelamente se buscar estudiar la sensibilidad del programa y en particular de los métodos de filtro de Kalman en un escenario donde hay pérdida de información.

1.4. Objetivos específicos

- Mejorar proceso existente sobre la manipulación de los archivos requisito del programa original, debido a que actualmente existen problemas de *hard-coding* respecto de la ubicación de éstos, así como ciertos reparos durante el proceso de selección de los mismos. Además se necesita reformular la configuración de parámetros funcionales de entrada (propios de la ejecución del programa, como los umbrales) para flexibilizar la entrada de estos.

- Generar una familia de métodos en términos de programación orientada a objetos empleando el patrón de diseño *Strategy* que implemente la familia de métodos conocido como Filtros de Kalman.

- Agregar una nueva variante de filtro que permita el uso de un modelo no lineal.

- Implementar gráficas de resultados en las que sea posible observar la curva generada por el flujo y la velocidad de flujo estimados por los filtros indicando la entropía aproximada del espacio de estados generado, visualizar la evolución de las mediciones y estimaciones tanto en secuencias de imágenes estampillas (pixéles) como en gráficas.

- Estudiar los resultados obtenidos con esta nueva versión del programa con cada uno de sus filtros sobre un conjunto de datos brindado por HiTS (del año 2015), a la par que el desempeño (medido en términos de uso de memoria principal y tiempo de ejecución) de éste. Por último se busca establecer un contraste con la versión original.

1.5. Organización de la tesis

Este documento contiene los siguientes capítulos: en el Capítulo 2 se describe en qué consiste una supernova, el proyecto HiTS 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).

En el Capítulo 3, se estudian tanto el desempeño como 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 4 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 5 se exponen los pasos del desarrollo del nuevo filtro además de la nueva funcionalidad relacionada con la carga de resultados previamente guardados.

Posteriormente, en el Capítulo 6, se exponen los resultados obtenidos para el desempeño de la nueva versión del programa (para los dos filtros originales) además de la cantidad de supernovas redescubiertas y el período en que se detectaron. El capítulo concluye con el análisis de los resultados y una comparación con aquellos obtenidos en el Capítulo 3.

Finalmente en el Capítulo 7 se presentan las conclusiones desprendidas de este trabajo y se propone el trabajo a futuro que se extiende de él.

2 | Antecedentes

En el presente Capítulo se exponen conceptos cruciales para la comprensión y desarrollo de este trabajo. Se describe en particular los objetos de interés que estimularon el desarrollo del survey HiTS: las supernovas de tipo II y el evento de shock-breakout asociado a éstas y una breve descripción de los datos obtenidos por este proyecto durante el año 2015. De la misma forma, se describe la base matemática en la que se basan las diferentes versiones del filtro de Kalman implementadas en el software original, es decir, los filtros de Kalman *básico* y de *máxima correntropía* así como una nueva versión del método conocida como *unscented* que permitiría la introducción de modelos no lineales en el proceso de filtrado.

2.1. Supernova tipo II

Una supernova de tipo II corresponde a un evento estelar con el que finaliza la vida de una estrella masiva (aquellas que en su proceso de formación poseen una masa superior a 10 masas solares [1]). Al 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, los cuales caen al núcleo de la estrella por su propio peso. Ver Figura 2.1.), y si la presión degenerada[2] de los electrones del plasma de la estrella no es suficiente para soportar este peso, la estrella se contrae abruptamente incrementando la temperatura de su centro a 10^{10} K.

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 capas más externas de la estrella son expulsadas en una gran explosión. Este fenómeno se denomina *supernova* e implica un aumento repentino del 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

[1] $M_\odot = (1.98847 \pm 0.00007) \times 10^{30}$ Kg
[2] presión que viene del principio de exclusión de Pauli

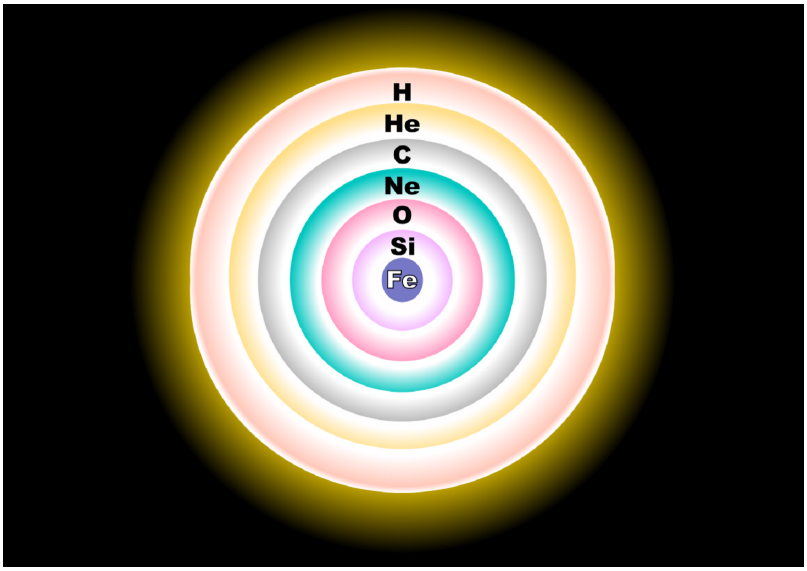


Figura 2.1: Esquema no a escala de la estructura de una estrella masiva previo a su explosión como supernova. Los elementos más pesados se alojan en el centro, mientras que los más livianos, como el hidrógeno o el helio lo hacen en la capa más externa. *Imagen publicada por R. J. Hall en WikiMedia Commons, 15 de agosto de 2007.*

rápidamente los primeros días (u horas), alcanzando un máximo, para luego decaer. Cabe destacar que existen otros tipos de supernova, cómo las de las de tipo Ia 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.

Las supernovas de tipo Ia 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 período de decaimiento suavizado (figura 2.2). Otra forma de diferenciarlas es la presencia de trazas de hidrógeno en los espectros de éstas: las supernovas de tipo Ia prácticamente no presentan hidrógeno (líneas de absorción distintivas) a diferencia de las de tipo II.

2.2. High Cadence Transient Survey: HiTS

El High Cadence Transient Survey [8] (desde ahora, HiTS) es un survey cuyo objetivo principal es detectar y seguir fenómenos transitorios 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

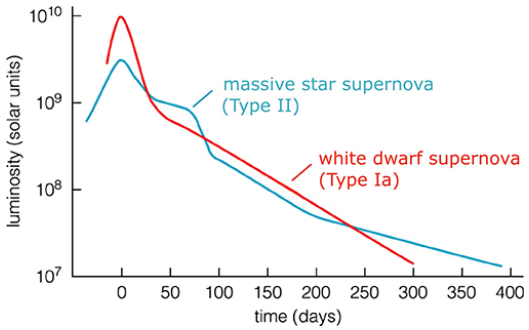


Figura 2.2: Curvas típicas de supernovas Ia (enana blanca) y II (estrella masiva). ©*2004 Pearson Education Inc., publishing as Addison-Wesley The Bizarre Stellar Graveyard.*

supernova II). Ver figura 2.3[3].

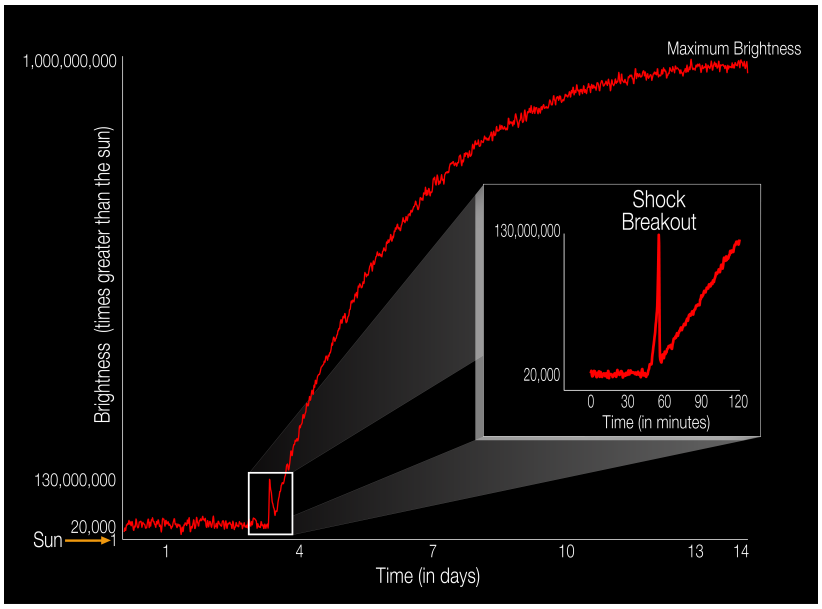


Figura 2.3: Diagrama que ilustra la evolución del brillo de una supernova en términos de luminosidad solar ($L_\odot$) durante días. Se resalta el fenómeno de *shock-breakout* apenas comienza el incremento de la luminosidad de la supernova. Esta imagen fue publicada en la página de la NASA destacando la primera vez que un evento como este es *capturado* en la banda visible (por el telescopio espacial Kepler). *NASA Ames/W. Stenzel. 2016*

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 (ver Figuras 2.4 y 2.5 dónde se muestra disposición de las cámaras CCD en el telescopio).

Durante el proyecto HiTS se realizaron tres campañas de observación en los años 2013, 2014

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

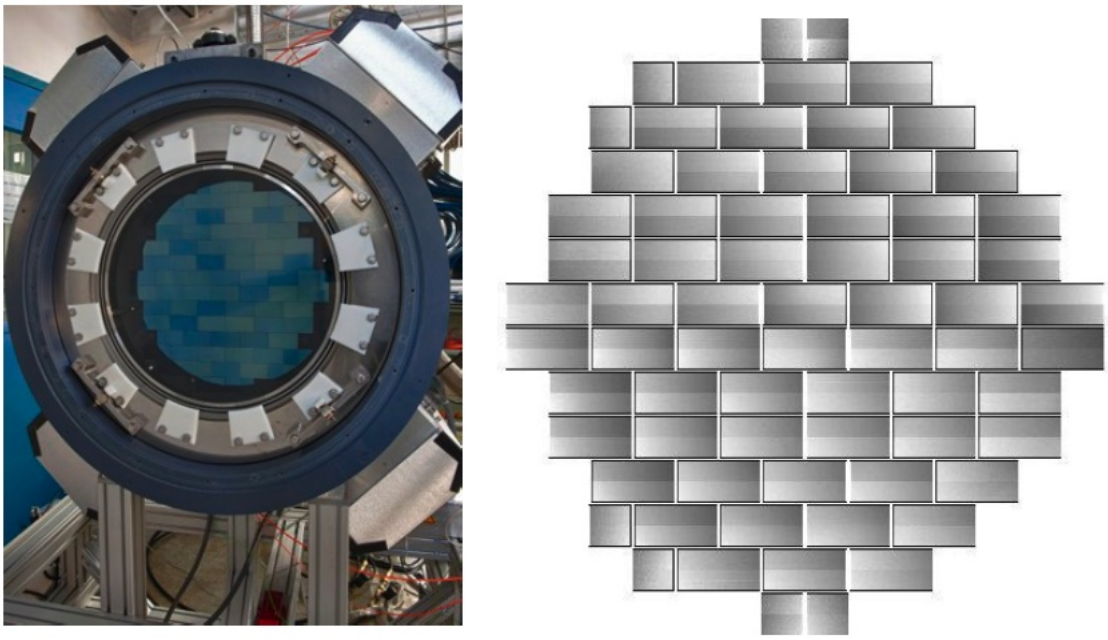


Figura 2.4: A la izquierda, estructura de la cámara DECam poblada con 62 chips CCDs. A la derecha, imagen *flat field* desde DECam. *Imágenes tomadas desde la página del Dark Energy Survey (www. darkenergysurvey. org/ the-des-project/ instrument/ the-camera).*

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. En estas campañas se obtuvieron más de 120 candidatos a supernova. Sin embargo, no se logró encontrar en estas rastros de SBO (figura 2.3) evento que comprendió uno de los principales objetivos de HiTS.

2.2.1. Datos obtenidos durante el año 2015

Como se mencionó anteriormente, la campaña del año 2015 tuvo lugar el primer semestre de ese año. El período en que se llevó a cabo fue durante los meses de febrero y marzo; específicamente entre los días 17 de febrero a 14 de marzo.

El período comprendido por los días 17 a 22 de febrero fue el de mayor latencia, obteniéndose cinco épocas (observaciones) por noche por cada campo. Posteriormente la toma de observaciones cesa y se reanuda a partir del 24 de febrero con una latencia de a lo más tres épocas por noche y campo con interrupciones de hasta diez días finalizando el 14 de marzo.

2.3. El filtro de Kalman

La evolución determinística de un sistema físico en el tiempo es conocida si el estado del sistema es medido con absoluta precisión en cada instante de tiempo (i.e., en un entorno donde es posible

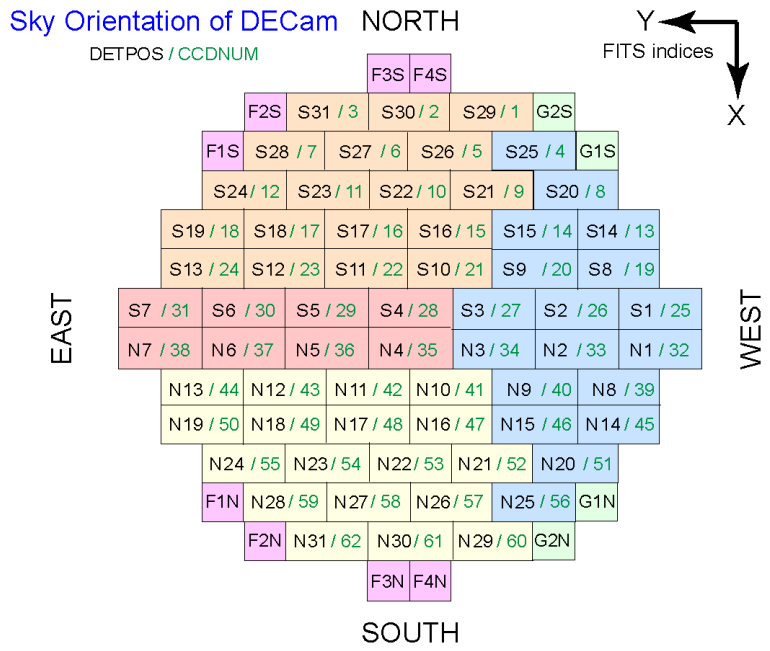


Figura 2.5: Orientaciones sobre el cielo y la huella espacial del arreglo de detectores en el plano focal. Se destacan los CCD cuya etiqueta comienzan con S o N, ya que estos corresponden a los detectores encargados de obtener las imágenes científicas. El etiquetado de estas componentes puede ser engañoso debido a que las iniciales de norte (North) y sur (South) están invertidas en relación a la orientación del cielo. *Imagen publicada en el sitio del CTIO (http://www.ctio.noao.edu/noao/node/2250).*

despreciar fenómenos cuánticos). Sin embargo toda medición está sujeta a incertezas finitas. Para sistemas los cuales son observados entre intervalos prolongados de tiempo, se prevee que las diferencias 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 puedan ser considerados confiables en un lapso de tiempo apropiado.

Los filtros de Kalman son métodos que proveen un compromiso (o 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 estudia el

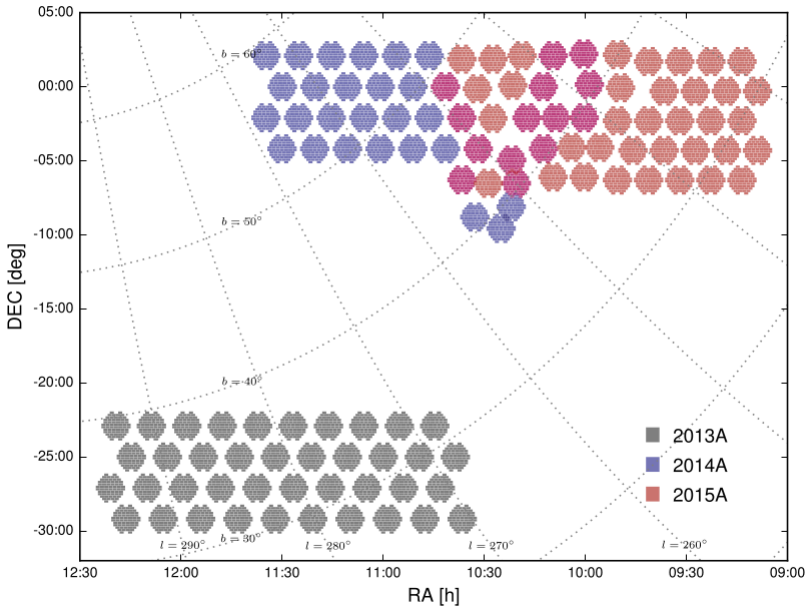


Figura 2.6: Distribución espacial de los campos observados durante los primeros semestres de los años 2013 (gris), 2014 (azul) y 2015 (naranjo). En tono rojo, los mismos campos del año 2015 y 2014 (superposición). *F. Förster et al., 2015. HiTS real-time supernova detections.*

comportamiento físico de algún sistema.

2.3.1. Filtro de Kalman Básico

El filtro de Kalman Básico [11] asume un comportamiento de sistema lineal y que las mediciones y las predicciones siguen una distribución Gaussiana.

A continuación se describen las componentes del 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.

Se hará uso de la notación en subíndices $m|n$, en las estimaciones de estado y covarianzas, para explicitar el 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* (vector de dimensión N).
- $P_{k|k-1}$: Matriz de covarianza *a priori* (matriz de dimensión $N \times N$).

Luego, en la fase de corrección se calculan:

- $\hat{z}_k$: Predicción de la medida.
- $\tilde{z}_k$: Residuo de la diferencia entre la medición real, z_k, y su predicción, $\hat{z}_k$.
- $\hat{x}_{k|k}$: 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 del algoritmo de Kalman básico (un esquema de los cálculos involucrados en el proceso de estimación de estados se puede ver en la figura 2.7):

1. **Fase predictiva:**

   Las ecuaciones de estimación de estado y matriz de covarianza *a priori* son:

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

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

2. **Fase correctiva:**

   Las ecuaciones de stimación de estado y matriz de covarianza *a posteriori* son:

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

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

   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 (S_k), con la que se calcula la ganancia de Kalman: K_k . Formalmente,

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

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

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

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

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

La figura 2.7 resume el proceso de predicción (obtención de las cantidades a priori, $\hat{x}_{k|k-1}$ y $P_{k|k-1}$) y corrección (generación de las estimaciones, $\hat{x}_{k+1|k}$ y $P_{k+1|k}$, a partir de las cantidades a posteriori $\hat{x}_{k|k}$ y $P_{k|k}$) del filtro de Kalman Básico.

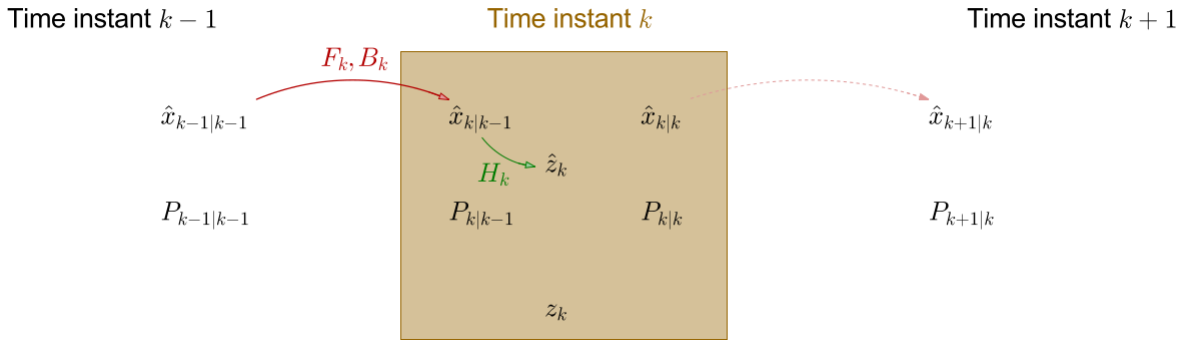


Figura 2.7: Representación del proceso de predicción (obtención de cantidades a priori) de las cantidades $\hat{x}_{k|k-1}$ y $P_{k|k-1}$; y de corrección (estimación a posteriori) para obtener las cantidades $\hat{x}_{k+1|k}$ y $P_{k+1|k}$. *E. Matsinos, 2016. The Kalman Filter: a didactical overview.*

2.3.2. Filtro de Kalman de Máxima Correntropía

El filtro de Kalman basado en máxima correntropía [5], 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. En esta oportunidad se utiliza el *criterio de máxima correntropía* para el proceso de corrección.

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, [9]). Para este filtro se emplea una función de kernel Gaussiana, dado por

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

La *función de costo* de máxima correntropía se define como:

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

La Ecuación 2.11 representa la función a maximizar, la que 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 $\tilde{e}_i$ estimado como,

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

Con estos residuos definimos las matrices diagonales, descritas en las Ecuaciones siguientes:

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

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

La matriz 2.14 corresponde a evaluaciones del kernel en errores estimados de predicción y la matriz 2.15 en errores propios de las observaciones (ruido). Luego se realiza una transformación de la covarianza del ruido de las mediciones usando una descomposición de Choleski ($B_{r,k}$) en el instante k [6], según la expresión:

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

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

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

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 estimación 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 2.13 a 2.19 se repiten secuencialmente hasta satisfacer la condición:

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

El valor de ϵ es definido por el usuario y corresponde a un criterio de detención (el algoritmo también puede detenerse definiendo un máximo de número de pasos). Finalmente se calcula la matriz de covarianza para el instante actual, k.

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

2.3.3. Filtro de Kalman Unscented (UKF)

Este filtro corresponde a aquel que se pretende agregar a la familia de filtros ya desarrollada en el programa base.

1. **Fase predictiva:**

   En esta versión del filtro [15] 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 puntos representativos alrededor de $\hat{x}_{k-1|k-1}$ y evaluar estos en la función no lineal f, para obtener las estimaciones de $\hat{x}_{k|k-1}$ y $P_{k|k-1}$. Estos puntos se conocen como *puntos sigma* (en la Figura 2.8 se visualiza el proceso de predicción al momento de obtener el primer conjunto de *puntos sigma*, propagarlos usando la función f y obtener $\hat{x}_{k|k-1}$ y $P_{k|k-1}$, es decir, la matriz de estados y de covarianza *a priori*).

   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:

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

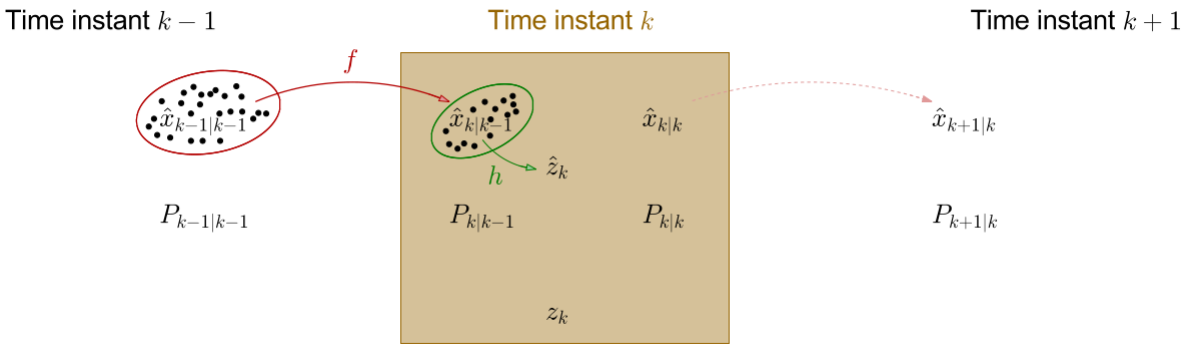


Figura 2.8: Representación del funcionamiento del filtro UKF. En esta oportunidad se hace uso de la función f y h para obtener las transformaciones $x_{k-1} \to x_k$ y $x_k \to z_k$. Esto se logra con la evaluación de los 2N+1 *puntos sigma* generados durante la etapa de predicción (y posteriormente en la etapa de corrección). *E. Matsinos, 2016. The Kalman Filter: a didactical overview.*

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

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

La matriz (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: $\{w_x^i\}$ y $\{w_p^i\}$. 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. Estos pesos son definidos como:

$$\begin{aligned} w_x^0 &= \frac{\lambda}{N+\lambda} \\ w_p^0 &= w_x^0 + 1 - \alpha^2 + \beta \\ w_x^i &= w_p^i = \frac{1}{2(N+\lambda)} \\ \sum_i^{2N} w_x^i &= 1 \end{aligned} \quad . \tag{2.24}$$

De 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 κ según la expresión siguiente:

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

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. La ecuación correspondiente es:

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

- Estimación a priori de la matriz de covarianza. La ecuación correspondiente es:

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

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.

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

La cantidad ψ_i representa la 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:

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

- Los residuos de las mediciones pueden obtenerse como:

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

- La matriz de innovación:

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

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

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

- 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 que 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 hizo uso de uno de los nodos del clúster de Leftraru, el supercomputador del NLHPC disponible para la comunidad investigadora desde el 2014 en 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

Para la ejecución de las pruebas se emplearon cuatro cores, 2400 MB por cada CPU (máxima RAM permitida) y un *job-array* de largo 93.

3 | Evaluación 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, a la par de estudiar los resultados en el proceso de detección de las 93 supernovas halladas en HiTS en la campaña del año 2015.

Por tanto se realizaron dos tipos de prueba: la primera, orientada a medir tiempo de ejecución y uso de memoria principal, se llevó a cabo en un computador personal de 8 GB (DDR4) de RAM sobre tres series de datos definidos por pares de CCD y campo en los que se sabe que hay una supernova. El segundo tipo de prueba se realizó en Leftraru, pasando por algunas complicaciones relacionadas con el diseño del programa ya que el código original contiene demasiadas líneas con el comando `glob`[1] de Python el cual lista reiteradamente el contenido de los directorios del home del usuario lo que finalmente termina saturando el nodo destinado para el lanzamiento del programa si se tiene una gran cantidad de archivos. Este problema pudo resolverse eliminando documentos resultantes de pruebas pristinas. Cabe destacar que se adaptó el programa para Python 3.6, ya que originalmente estaba para 2.7 (ambos tipos de pruebas se realizaron para la versión 3.6).

La pipeline original (`candidate_search`) está estructurada por un proceso que puede dividirse en dos bloques: el primero está destinado a buscar una supernova confirmada por HiTS (cuya coordenada es conocida) y encontrar nuevos candidatos; mientras que el segundo bloque, con la lista de coordenadas de nuevos candidatos, está destinado a obtener la información de estos. Sin embargo, nos encontramos con un segundo problema de implementación: la existencia de código duplicado. Ya que ambos bloques están repitiendo los mismos pasos salvo el último en el que se diferencian en el registro de los resultados: en uno se guarda la información de la supernova conocida (sólo si la detecta) y en el siguiente, sólo si hay nuevos candidatos, guarda la información de estos.

El proceso de detección se inicializa en una instancia de RunData, **listando los archivos desde disco**(apoyándose de la clase **FITSHandler**), para preparar las imágenes y el resto de

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

archivos que serán usados para el proceso iterativo que se lista a continuación:

1. **Cálculo de flujo:** A partir de las imágenes científicas y de calibración se obtiene el flujo (en ADU[2]) por pixel y es registrado en una matriz (`numpy array`) en cada iteración.

2. **Proceso de estimación con filtro de Kalman:** En este paso se realizan los procesos de predicción y corrección para obtener una nueva estimación (se usa alguna instancia de uno de los filtros **KalmanFilter** o **MaximumCorrentropyKalmanFilter**).

3. **Detección de fuentes:** En este paso se estudia la idoneidad de los pixeles tanto del flujo como de las estimaciones de los mismos (obtenidos con el filtro de Kalman) y del resto de las imágenes al momento de verificar una serie de criterios con los cuales estos se agruparían entre vecinos para formar potenciales candidatos (este proceso se realiza en **SNDetector**).

4. **Actualización de candidatos:** Revisa si hay nuevos candidatos a ser considerados. Se registran en un arreglo (`numpy array`) y guarda resultados de la supernova conocida en caso de encontrarse en un arreglo de diccionarios (variable de instancia *obj* de un objeto de la clase **Observer**).

Si en el proceso anterior se encuentra algún candidato, se procede a repetir los pasos de obtención de flujos, estimaciones y filtrado de pixeles. La diferencia está en que en esta ocasión se van guardando los resultados en una estructura de datos de diccionario. El nuevo ciclo queda como sigue:

1. **Repetición de los pasos anteriores 1-3**

2. **Guardado de resultados:** La información de los nuevos candidatos encontrados previamente es guardado en la lista diccionarios *obj* (variable de la clase **Observer**) y registrado en disco usando el formato NPZ (formato brindado por Numpy para comprimir datos).

El diagrama de la Figura 3.1 entrega una perspectiva general de la secuencia de pasos que realiza el programa.

Cabe destacar que durante el proceso de refactoring del programa original se encontró que la lista de archivos, que debe ser procesada en orden de acuerdo a la época (o fecha de observación) no estaba siendo bien ordenada durante el proceso de selección de imágenes científicas a partir del valor de `airmass` en la clase FitsHandler.

[2] Analog-to-digital unit

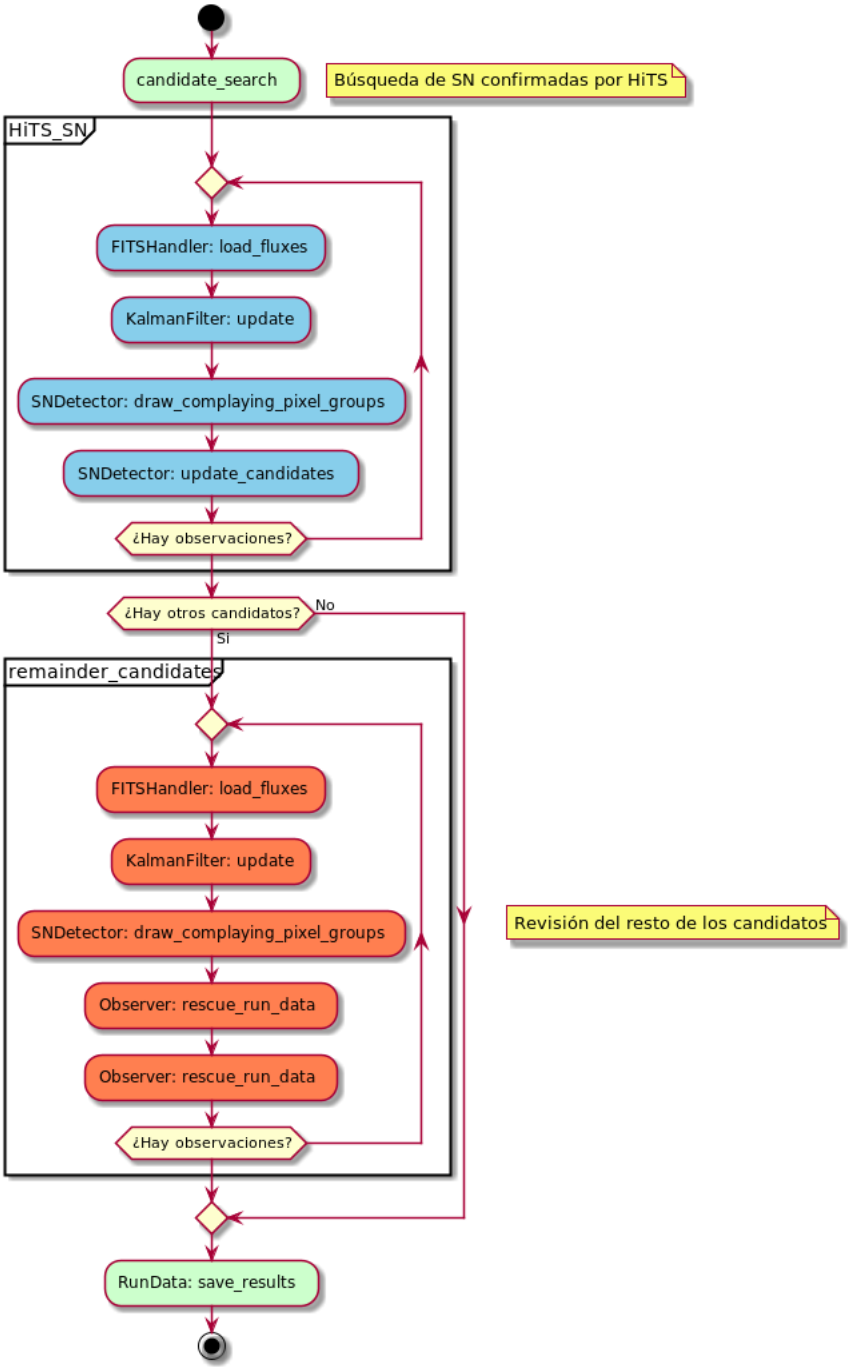


Figura 3.1: Diagrama de flujo del programa original. Se aprecían dos ciclos principales: el primero está destinado a la búsqueda de una supernova de HiTS, y el segundo a la revisión de la lista de posibles candidatos encontrados durante la verificación de la supernova de HiTS. Notar que hay pasos que se repiten en la realización de ambos análisis.

3.1. Estructura de datos

Debido a la naturaleza de la información de entrada (imágenes) se debe trabajar en píxeles, por lo que la estructura de datos que representen las variables de estado debe considerar la dimensión

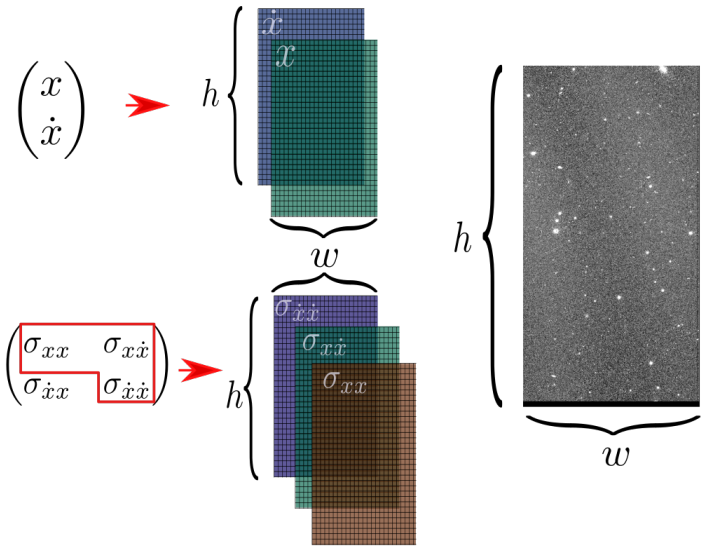


Figura 3.2: Esquema de las estructuras de datos usadas para la representación de las matrices de estados y de las matrices de covarianza de la relación entre el flujo (x) y la velocidad de flujo ($\dot{x}$) para cada pixel, de una imagen de ancho w y altura h. Para los conjuntos de datos, $w = 2046$ y $h = 4096$.

de las imágenes (considerando que por cada píxel se debe modelar las variables de estado flujo y velocidad de flujo, y covarianza asociada). Debido a esto, en el trabajo de Pablo Huentelemu [14] se diseñaron *hiperrectángulos* para modelar las variables de estado y covarianza de todos los pixeles de una imagen. Los datasets contienen imágenes de dimensión 2046×4096. La figura 3.2 muestra el esquema de sistema de matrices o *hiperrectángulos* usada para representar el estado de cada píxel y su respectiva covarianza.

3.2. 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 realizaron 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, 23 y 18 observaciones respectivamente.

Las tablas 3.1 y 3.2 muestran el tiempo en segundos que toma el proceso de detección de candidatos. Se destacan como procesos separados el reconocimiento de la supernova de HiTS y posteriormente, el proceso de estudio de los nuevos candidatos, respectivamente, empleando para ambos procesos el filtro Básico.

Tabla 3.1: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el período de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman Básico. La última fila corresponde a la media por observación.

ID	**Cálc. Flujos [s]**	**Aplic. KF [s]**	**Agrup. Pixeles [s]**	**Actual. Candidatos [s]**
SN14	293.91	24.90	68.30	0.02
SN18	260.09	22.56	45.52	0.00
SN80	204.93	17.69	38.21	0.00
$\bar{t}/Obs$	11.00	0.97	2.24	0.00

Tabla 3.2: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el período de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman Básico. Se observa que para las dos últimas supernovas los tiempos son cero ya que no se encontraron más candidatos.

ID	**Cálc. Flujos [s]**	**Aplic. KF [s]**	**Agrup. Pixeles [s]**	**Guardar resultados [s]**
SN14	303.18	27.28	72.34	0.02
SN18	0.00	0.00	0.00	0.00
SN80	0.00	0.00	0.00	0.00
$\bar{t}/Obs$	11.66	1.05	2.78	0.00

Las tablas 3.3 y el 3.4 describen el tiempo (en segundos) tomado tanto para el proceso de detección de la supernova de HiTS y el posterior procesamiento de posibles nuevos candidatos (en ese orden) usando el filtro de máxima correntropía.

Tabla 3.3: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el período de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman de Máxima Correntropía.

ID	**Cálc. Flujos [s]**	**Aplic. KF [s]**	**Agrup. Pixeles [s]**	**Actual. Candidatos [s]**
SN14	342.44	798.48	76.47	0.00
SN18	273.64	566.89	47.96	0.00
SN80	210.68	420.12	36.05	0.00
$\bar{t}/Obs.$	12.25	26.23	2.34	0.00

La tabla 3.5 muestra el tiempo total comprendido por ambos subprocesos usando el filtro de Kalman Básico. Se destaca la diferencia del consumo de tiempo ante la ausencia y presencia de nuevos candidatos a supernova. Por otro lado, la tabla 3.6 muestran las mismas medidas de tiempo para el filtro de máxima correntropía. Se destaca el aumento considerable de tiempo al usar este último filtro en relación al primero.

Tabla 3.4: Resultados de tiempos de ejecución correspondientes a calculo de flujos, estimación de los filtros, agrupación de pixeles y filtrado de los mismos durante el período de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman de Máxima Correntropía. Se observa que para las dos últimas supernovas los tiempos son cero ya que no se encontraron más candidatos.

ID	**Cálc. Flujos** [s]	**Aplic. KF** [s]	**Agrup. Pixeles** [s]	**Guardar resultados** [s]
SN14	307.64	680.65	65.67	0.02
SN18	0.00	0.00	0.00	0.00
SN80	0.00	0.00	0.00	0.00
$\bar{t}/Obs.$	11.83	26.18	2.53	0.00

Tabla 3.5: 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. La última fila corresponde a tiempo total promedio por observación.

ID	**Búsqueda SN** [s]	**Revisión candidatos**[s]	**Tiempo total** [s]
SN14	387.13	402.82	789.95
SN18	328.17	0.00	328.17
SN80	260.83	0.00	260.83
$\bar{t}/Obs.$	14.55	–	–

Tabla 3.6: 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.

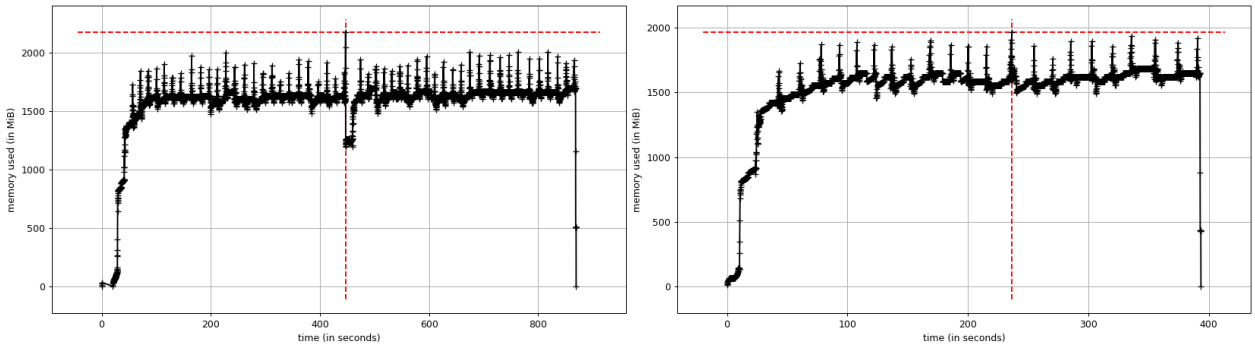
ID	**Búsqueda SN** [s]	**Revisión candidatos** [s]	**Tiempo total** [s]
SN14	1217.39	1053.98	2068.80
SN18	888.49	0.00	0.00
SN80	666.85	0.00	0.00
$\bar{t}/Obs.$	40.83	–	–

3.3. 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[3].

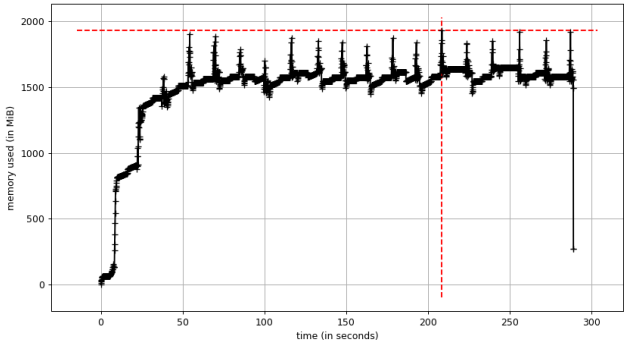
La figura 3.3 muestra el comportamiento del uso de memoria principal (en términos de Mebibytes) para los tres conjuntos de datos empleados, usando el filtro de Kalman Básico. Se distingue un uso más intensivo (máximo alcanzado) durante el proceso en que se usaron los datos de la SN14, la cual no sólo posee una secuencia de imágenes mayor (de 26) sino también corresponde a aquella en donde se encontraron posibles nuevos candidatos. En la tabla 3.7 se expone el máximo consumo generado durante la ejecución de la pipeline empleando el filtro Básico.

[3] $1MiB \simeq 1.049MB$



(a) *Memoria ocupada en SN14*

(b) *Memoria ocupada en SN18*



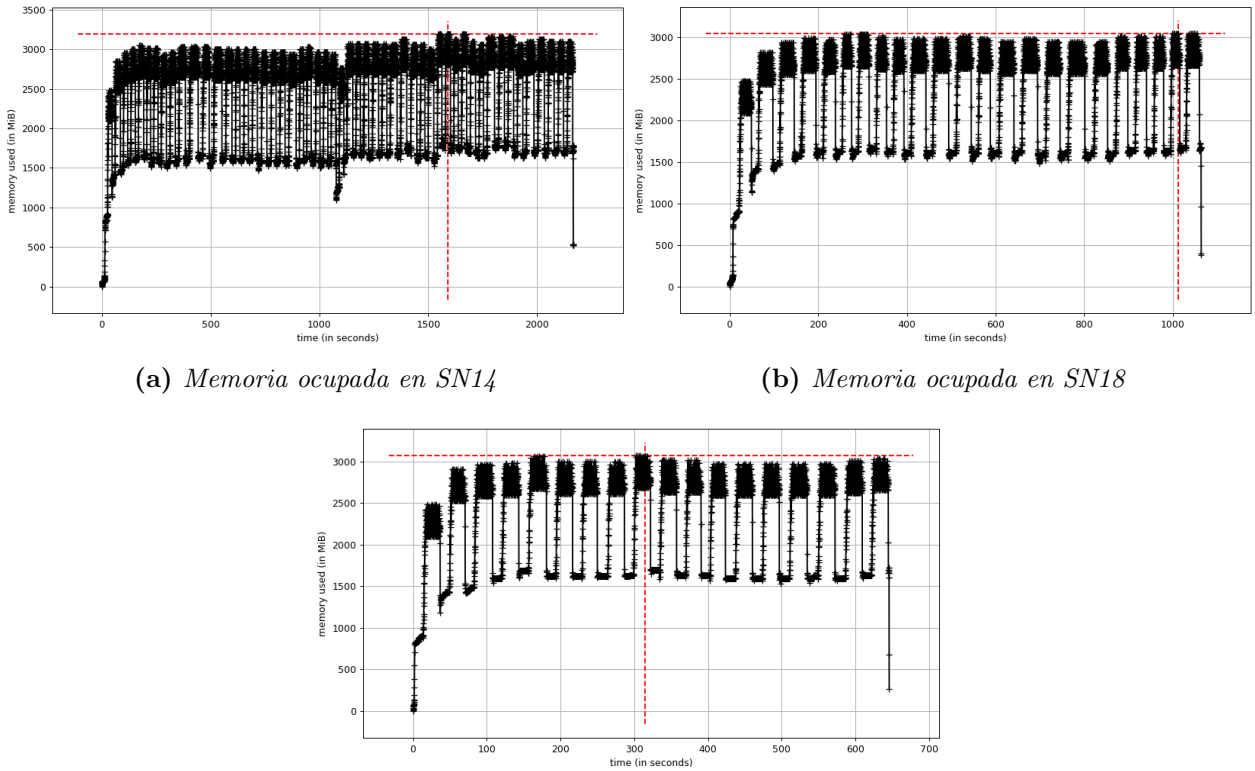
(c) *Memoria ocupada en SN80*

Figura 3.3: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos de las supernovas SN14, SN18 y SN80, usando el filtro de Kalman Básico.

Tabla 3.7: Memoria principal (en unidades de MB) usada durante la ejecución del programa original con la versión básica del filtro de Kalman.

ID	Memoria [MB]
SN14	2282.42
SN18	2063.02
SN80	2021.27

La figura 3.4 muestra el consumo de memoria (en Mebibytes) para la ejecución del programa original para los tres conjuntos de datos, usando el filtro de máxima correntropía. Se destaca un comportamiento similar al obtenido usando el filtro básico debido a la detección de posibles candidatos con el conjunto de datos de la SN14. Por otro lado, se desprende un mayor consumo de parte de este filtro en relación a la versión básica. La tabla 3.8 describe el consumo máximo de memoria alcanzado con el uso del filtro de máxima correntropía.



(a) *Memoria ocupada en SN14*

(b) *Memoria ocupada en SN18*

(c) *Memoria ocupada en SN80*

Figura 3.4: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos de las supernovas SN14, SN18 y SN80. En los tres lanzamientos se usó el filtro de Kalman de correntropía máxima.

Tabla 3.8: Memoria principal (en unidades de MB) usada durante la ejecución del programa original usando filtro de Kalman de máxima correntropía.

ID	Memoria [MB]
SN14	3353.13
SN18	3194.96
SN80	3209.67

3.4. Detección

Se realizaron las pruebas de detección sobre los conjuntos de datos de las 93 supernovas en Leftraru durante el mes de febrero de 2019, usando los umbrales de 200.0 para estimación de flujo y 50.0 para la velocidad de flujo. En el caso del filtro de máxima correntropía se usó el mismo valor de σ empleado para el filtro básico: 1000.0. La Tabla 3.9, muestra el número de detecciones exitosas como verdaderos positivos (TP), detecciones fallidas como falsos negativos (FN), falsos positivos (FP) y escenarios en donde la ejecucı'on del programa falló (por no encontrar archivos necesarios).

Para los parámetros usados (párrafo anterior) se obtienen resultados idénticos para ambos filtros

Tabla 3.9: Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) encontrados usando cada uno de los filtros. La cuarta columna, **NaN** indica el número conjunto de datos que no se pudieron procesar por falta de algún(os) archivo(s). No se observan diferencias sustanciales entre los resultados de cada filtro.

Filtro	**TP**	**FN**	**FP**	**NaN**
Básico	35	55	53	3
MCC	35	55	52	3

en términos de cantidad de supernovas detectadas. En ambos casos se detectaron 35, sin embargo existe una leve diferencia en la cantidad de falsos positivos: el programa, usando el filtro de máxima correntropía encuentra un falso positivo menos que al usar el filtro básico. Para ambos casos, tres secuencias de supernovas no pudieron ser procesadas (cuyos resultados fueron etiquetados como *NaN*) debido a la falta de imágenes de invarianza inversa (esenciales para el cálculo de flujo).

Cabe destacar que durante las primeras pruebas con el programa original, en su versión para Python 2.7, se detectaban 33 supernovas de las 93, no obstante al actualizar el código para una versión compatible con Python 3.6, se obtiene un mejor resultado para los dos tipos de filtros, detectándose dos supernovas más. Se piensa que esto puede deberse a la aactualización de la librería PyMorph[4] por Mahotas[5] (ambas librerías desarrolladas por el mismo autor, Luis P. Coelho[6]).

3.4.1. Observaciones

A partir de los resultados obtenidos, se procedió a estudiar las curvas de luz (en términos de ADU[7] considerando escenarios en donde supernovas conocidas son detectadas por el programa (con ambos filtros) (figura 3.5) y donde no ocurre (figura 3.6).

En la figura 3.5 se observa, a grandes rasgos, la presencia una etapa de crecimiento en el flujo de los objetos (supernovas), coincidiendo con el período de alta cadencia mencionado en el capítulo anterior (Capítulo 2, subsección 2.2.1), mientras que por el contrario en la imagen 3.6 el comportamiento de los flujos es más bien irregular y no se distinguen etapas de crecimiento constante. Hay que mencionar que el algoritmo principal, aquel cuya tarea es la de filtrar y agrupar píxeles bajo criterios de descarte (implementado en **SNDetector**), emplea un sistema de *alerta* interno con el cual se gatilla una detección cuando un grupo determinado de pixeles satisface una serie de requisitos en cuatro épocas consecutivas. Para gatillar esta alerta, uno de los requisitos que debe

[4] https://pythonhosted.org/pymorph/
[5] https://mahotas.readthedocs.io/en/latest/
[6] https://github.com/luispedro
[7] analog-to-digital unit

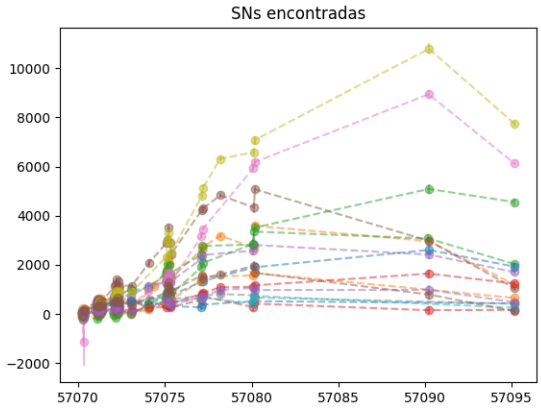


Figura 3.5: Curvas de luz (en ADU vs MJD) de 16 supernovas detectadas tanto por el filtro de Kalman básico como el de máxima correntropía.

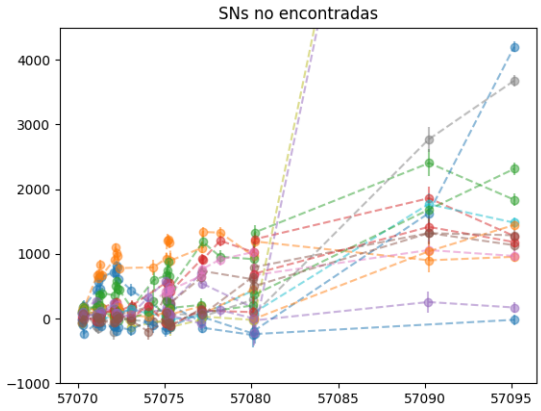


Figura 3.6: Curvas de luz (en ADU vs MJD) de 16 supernovas no detectadas ni por el filtro de Kalman básico ni por el de de máxima correntropía.

cumplir un grupo de píxeles para considerarse candidato a supernova es el crecimiento continuo de los flujos por píxel. Si ocurre un descenso entonces el grupo es inmediatamente descartado.

Las Imágenes 3.7 y 3.8 muestran dos casos del intérvalo de tiempo en que se realizan detecciones exitosas usando ambas implementaciones de los filtros del programa original (ver Apéndice, sección 7.1 para conocer lista completa de fechas de detecciones).

Falsos positivos

El poder reconocer falsos positivos a través de este algoritmo resulta complicado si no se cuenta con una herramienta externa de visualización o de machine learning (por ejemplo, usando un clasificador de conjuntos de pixeles que evalúe la probabilidad de que se trata efectivamente de una

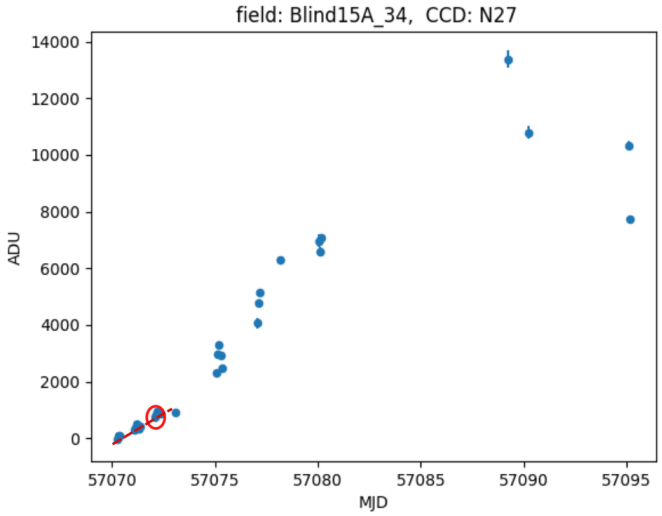


Figura 3.7: Curva de luz (en ADU vs MJD) de supernova registrada en el CCD N27, campo 34. La detección realizada por los filtros básico y de máxima correntropía se realizó en el MJD 57072.214 (época u observación 7 de 18).

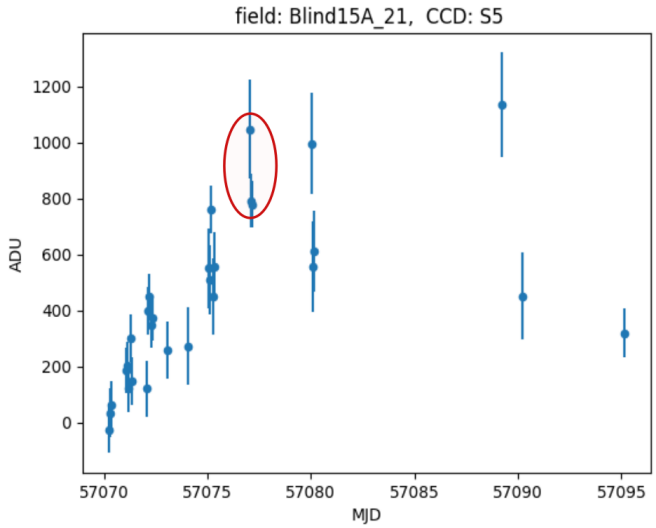


Figura 3.8: Curva de luz supernova observada en el CCD S5, campo 21. Ambos filtros detectaron la supernova en el MJD 57077.166 (época u observación 23 de 27).

supernova joven). Esto se debe a que existen variados objetos como estrellas variables o *artefactos* que corresponden a zonas ruidosas de la imagen científica (en particular, bordes) y pueden proveer valores alterados de flujo lo que en algunos casos puede conllevar a la detección de falsos positivos.

Una forma de poder estudiar un falso positivo es a través de la visualización de su espacio de fase y la obtención la entropía de la curva.

4 | Refactoring

El refactoring del código original se realizó en Python 3.6 (versión diferente a la que el programa fue implementado inicialmente: 2.7). Además se mantuvieron casi la totalidad de las librerías cambiando únicamente Pymorph por Mahotas, debido a que la primera dejó de ser mantenida desde el 2010 y corresponde a la librería para Python 3.

4.1. Manejo de datos de entrada

Toda las imágenes de entrada son manipuladas y servidas por 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.

El archivo que esta clase consume, para la configuración de las rutas, debe contener los siguientes campos (ver ejemplo B .2):

- `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).
- `psfDir`: Directorio donde se encuentran los modelos de psf (apéndice 1) 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 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`. Estos archivos están ubicados en el path `afluxDir`.
- `psfRegEx`: Expresión regular que describe el nombre de las imágenes que guardan el modelo de PSF en el directorio `psfDir`.

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 el Capítulo 5 de nuevas funcionalidades 5.

4.1.1. 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 strings que indiquen el semestre (`semester`), el campo (`field`) y el ccd (`ccd`) que se quiere analizar. Puede hacerse uso de los valores de las variables de instancia que la misma clase DataPicker recibe en su constructor. 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 (máscaras, imágenes 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 (Apéndice 2), seleccionando aquellas obtenidas en fechas cuyo valor de airmass calculado es menor a 1,7. De esta secuencia de imágenes científicas resultante se obtiene una lista de fechas que cumplen esta condición medidas en términos de *día juliano modificado* o *Modified Julian Date* (MJD) de tipo punto flotante. Estos valores son ordenados de forma creciente.

- `select_fits(dir)`

  Selecciona y ordena los elementos de la lista de imágenes de formato fits del directorio `dir` usando la lista de MJDs (guardada en la variable de instancia `mjd` de la clase) generada en `filter_science_images()` escogiendo sólo aquellas cuyas fechas correspondan a las fechas indicadas.

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

  Debido a que los archivos de extensión NPY no poseen la variable MJD en su estructura (en los archivos FITS encontramos este valor en el header de la imagen) deben de filtrarse de forma diferente. Para este caso el filtrado de este tipo de archivos se lleva a cabo a través de la revisión de sus nombres, ya que comparten patrones con los nombres de ciertas imágenes fits. Por ejemplo, los nombres de las imágenes de PSF, de formato NPY, poseen similitud con los nombres de las imágenes FITS de diferencia; igualmente los archivos `aflux` de formato NPY poseen parecidos en sus nombres con las imágenes científicas. Este parecido es medido a través de un substring diferente para cada tipo de archivo NPY, definido por la posición inicial `init_index`, en el nombre del archivo FITS y largo `rest_len`. `n_pos` indica la posición de un carácter específico " _ " en dicho substring para validar esta comparación.

4.2. Determinación de flujos

El cálculo del flujo, en este refactoring, se independizó del manejo de archivos (en el programa orginal estaba alojado en la clase FITSHandler) y se implementó en el script `utils` pensado como librería.

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

- `naylor_photometry(invvar, diff, psf)` [13]:
  Calcula el producto del flujo por su varianza. 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.

Estos métodos son llamados desde la rutina RoutineHandler (ver Capítulo 5).

4.3. 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 (ver familia de métodos resultante en la figura 4.1). 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 estado, la matriz de covarianza de estado, y las predicciones de las matrices de estado y covarianza determinadas en el paso anterior (con la finalidad de actualizar estas variables). Su firma queda como:

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

  Este método entrega finalmente las matrices de estado y covarianza de estado predicho.

- **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 estado 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)
```

Esta función entrega finalmente las matrices de estado y covarianza de estado corregido.

4.3.1. 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).

4.3.2. 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. El constructor de esta clase recibe los siguientes parámetros:

- `epsilon`: Cantidad con la cual se contrastará el error o precisión que se quiera lograr con la estimación.
- `max_iter`: Número máximo de iteraciones.
- `silverman`: *boolean*. Determina si se usa o no la apoximación de Silverman para el ancho de banda del kernel Gaussiano.
- `std_factor`: Factor de desviación estándar usado en la aproximación de Silverman.
- `sigma`: Sigma usado para el kernel Mercer.

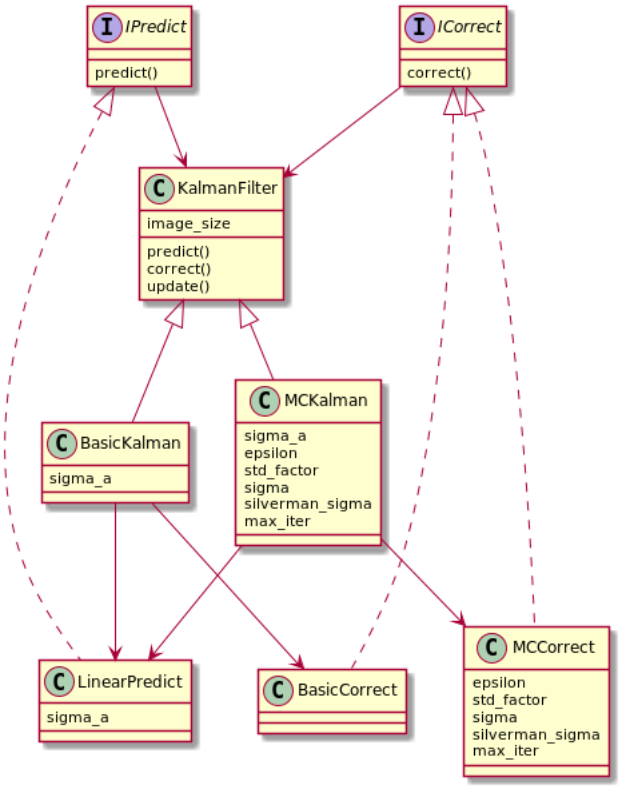


Figura 4.1: Familia de filtros de Kalman y patrón *strategy* usado en la implementación de los métodos `predict` y `correct`.

4.3.3. 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`.

4.4. Detección de candidatos

Mientras que la detección de candidatos en el programa original se realiza en una instancia de la clase SNDetector, la detección en la nueva versión se realiza en SourceFinder, el cual funciona de la misma forma que su antecesor. El cambio de nombre representa la intencionalidad de extender

la funcionalidad de esta clase no sólo a la detección de supernovas sino además encontrar objetos como estrellas variables.

Su constructor requiere los siguientes argumentos:

- `flux_thresh:` Umbral de corte para el flujo estimado por el filtro de Kalman.
- `flux_rate_thresh:` Umbral de corte de velocidad de flujo estimado por el filtro de Kalman.
- `rate_satu:` Tasa de saturación en la velocidad de flujo.
- `accum_neg_flux_depth:` Cantidad de épocas de registro de píxeles negativos (para la construcción de una matriz de booleans de esta profundidad).
- `accum_med_flux_depth::` Cantidad de épocas de registro de píxeles cuya intensidad mediana (durante las épocas) es mayor a 1500.
- `image_size:` Dimensión de las imágenes FITS.
- `n_consecutive_obs:` Número de alertas u observaciones consecutivas a considerar para confirmar una detección.

4.4.1. SourceFinder

La clase SourceFinder posee los siguientes métodos:

- `pixel_discard`:
  Método en el que se realiza el descarte de píxeles de forma individual, siguiendo los siguientes criterios:

  1. Si el flujo estimado por el filtro de Kalman para un pixel es menor que el umbral dado.
  2. Si la velocidad de flujo estimada por el filtro de Kalman es menor que el umbral de la velocidad de flujo multiplicado por un factor determinado la tasa de saturación en la velocidad de flujo y el flujo estimado por el filtro.
  3. Si un pixel de la imagen científica es menor a la mediana más cierto delta (en este trabajo, siguiendo la línea de desarrollo de P. Huentelemu [14], se consideró 5.0) es descartado.
  4. Si las varianzas de flujo son mayores a 150.0 (valor estimado por el autor del software original [14]).
  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 píxeles no caen en etiquetas de invalidación dentro de la máscara que ha sido procesada para marcar también los píxeles vecinos a los realmente defectuosos.
7. Si los píxeles no han caído dentro del descarte por superar la mediana estimada a partir de cuatro épocas.

- `grouping_pixels()`:

  Este método trabaja con las etiquetas determinadas en el paso anterior en un arreglo de matrices (`numpy array`) denominado `pixel_flags` (variable de instancia). Además recibe el índice de MJD correspondiente a la observación de tal fecha. La agrupación de píxeles se realiza gracias a funciones brindadas por la librería Mahotas, usando el método `label` para encontrar dominios cerrados en el mapa de píxeles validados.

- `filter_groups(science, flux, var_flux, state, base_mask)`:

  Este método recibe la imagen científica, el flujo y su varianza, el estado determinado por el filtro de Kalman y la máscara correspondiente a una época específica. El filtrado de grupos de píxeles se lleva a cabo bajo las siguientes reglas de descarte:

  1. Descarte de grupo por contener posible mala resta alrededor (valores negativos).
  2. Si no hay máximos locales dentro del grupo de píxeles encontrados dentro de la imagen científica.
  3. Si no hay máximos locales dentro del grupo de píxeles encontrados dentro de la matriz de flujo (calculado por `calc_fluxes`).
  4. Si no hay máximos locales dentro del grupo de píxeles encontrados dentro de la matriz velocidad de flujo.
  5. Si los valores de los píxeles 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 píxeles de cuatro observaciones consecutivas.
  9. Si la varianza del flujo del pixel del centro del grupo es mayor al determinado por el umbral.

- `update_candidates(mjd)`:

  En la estructura `CandData` se van registrando fechas (MJD) en que se han detectado candidatos

previamente o se han detectado por primera vez. Es una estrutura tipo lista en la que se van guardando diccionarios que contienen, cada uno, las coordenadas de un objeto, las épocas en las que ha sido detectado y si corresponde o no a una supernova conocida.

- `check_candidates(SN_index, SN_pos)`:
  Verifica que dado un índice de supernova (`SN_index`) y sus coordenadas `SN_pos` se ha detectado dentro de los candidatos encontrados.

- `draw_complying_pixel_groups(science, state, state_cov, base_mask, dil_mask, flux, var_flux, mjd)`:
  Este método es el que llama a `pixel_discard` para etiquetar píxeles para el descarte y no ser considerados en el paso de agrupamiento al llamar a `grouping_pixels`. Luego se invoca el método `filter_groups` para hacer el descarte a nivel grupal y obtener candidatos. La última llamada es para el método `update_candidates` para actualizar lista de candidatos encontrados en la variable de instancia `CandData`.

  Como argumentos recibe todos los elementos necesarios para ejecutar los métodos que llama.

4.5. Visualización de resultados

En el proceso de visualización participan dos clases: Observer y Visualizer. La primera clase es la encargada de generar una lista de diccionarios dentro de los cuales se almacena información de los candidatos encontrados durante el proceso de detección. Esta lista es guardada como una variable de instancia en Observer denominada `objects` y la información contenida por cada uno de los diccionarios corresponde a las siguientes componentes:

- Ubicación del objeto en la primera imagen científica procesada, como un arreglo de *floats* de largo dos.

- Lista de épocas en las que el objeto fue detectado.

- Listas de estampillas de diferente profundidad, de ancho y alto 21×21. Las estampillas de cada lista tendrán una profundidad propia dependiendo de la estructura que estén almacenando. Estas estructuras pueden ser imágenes como la científica, de diferencia, máscara, etc., o a estructuras como las etiquetas de píxeles grupales e individuales, matrices de estado, etc. Cada una de las estampillas de cada lista corresponderá a la medición, cálculo o estimación obtenida para una época específica. Además cada una de las estampillas está centrada en la coordenada del objeto candidato a revisar.

Para el registro de estos datos se emplean dos métodos de OBSERVER listados a continuación:

- `set_space (cand_data):`

  Recibe lista de candidatos (coordenadas) `cand_data` y crea lista de diccionarios, `objects`, para guardar la información de los primeros para todas las épocas analizadas creando arreglos (de la librería NUMPY) de estampillas 21×21 destinadas a guardar los datos de las diferentes estructuras: imágenes, matrices de estado estimado y predicho, covarianzas por pixel, matrices de etiquetas de píxeles, etc. Cada estampilla estará centrada en la posición del candidato en la primera imagen científica analizada.

- `look_candData(cand_data, pred_state, pred_state_cov, kalman_gain, state, state_cov, time_mjd, flux, var_flux, science, diff, psf, base_mask, dil_base_mask, pixel_flags, pixel_group_flags, mjd_idx):`

  Con el espacio generado en la estructura `objects`, se procede a ejecutar la pipeline principal (calculo de flujo, generación de estimaciones con filtro de Kalman y proceso de detección) para, en esta ocasión, ir guardando resultados de los candidatos en `cand_data` en su diccionario respectivo por cada época (cuyo índice está representado por el argumento `mjd_idx`). Por tanto recibe como argumentos las matrices de estados y covarianzas predichos (`pred_state` y `pred_state_cov` respectivamente), las matrices de estados y covarianzas estimados (`state` y `state_cov`) la matriz de ganancia de Kalman (`kalman_gain`), matriz de flujo calculado y varianza asociada (`flux` y `var_flux`), imagen científica (`science`) y de diferencia (`diff`), imagen de la PSF usada (`psf`), matrices de etiquetas de píxeles (`pixel_flags` y `pixel_group_flags`) e imagen de la máscara usada (base_mask) junto a su versión post-proceso de dilatación (dil_base_mask).

- `plot_results(semester, field, ccd, plots_path)`

  Finalmente, con la variable de instancia `objects` lista, es posible generar los gráficos de cada candidato con este método, el cual recibe como parámetros el semestre (`semester`), el campo (`field`) y el CCD (`ccd`) de donde se obtuvieron las imágenes junto a la ruta al directorio donde se quiere guardar las imágenes (`plots_path`) como argumentos. Los gráficos son generados por una instancia de la clase VISUALIZER.

La clase VISUALIZER permite la obtención de tres tipos gráficas de acuerdo al método llamado.

- **Estampillas**

  Se muestra el comportamiento de los píxeles en las estampillas de dimensión 21×21 de las siguientes estructuras: imagen científica, PSF, flujos observado y su varianza, estados de flujo y su velocidad estimados por filtro de Kalman, etiquetas grupales e individuales de píxeles.

- **Curva de estado**

  Esta curva se logra a partir de los valores estimados de flujo y de las velocidad de esta obtenidos por el filtro de Kalman. Esta gráfica muestra la complejidad de la curva visualizada calculando su entropía [3].

Estas gráficas son generadas gracias a los siguientes métodos de VISUALIZER:

- `print_lightcurve(obj, obs_rad, figsize1, figsize2, save_filename):`

  Este método está destinado a crear series de tiempo, para contrastar, de diferentes variables de interés tales como el flujo observado, estimado y predicho (medidos en el pixel central del candidato) visualizadados en la misma gráfica. Del mismo modo, en un gráfico dispuesto bajo el primero, se dibuja las series de tiempo de las velocidades estimadas y predichas. Posteriormente se grafica la evolución de las etiquetas de píxeles individuales y grupales, y el último gráfico generado corresponde a la serie de tiempo de las varianzas y covarianzas de las diferentes variables como flujo, estimaciones de flujo y sus predicciones obtenidas con el filtro de Kalman, etc.

  Los parámetros `figsize1` y `figsize2` corresponden a la altura y ancho de la imagen generada. Por último, `save_filename` corresponde al nombre con el que se guardará el archivo en disco en formato PNG. Ejemplo de imagen resultante en figura 4.2.

- `print_stamps(obj, figsize1, figsize2, save_filename):`

  Recibe como entrada un diccionario de la lista de objetos almacenados por una instancia de OBSERVER. Con esta función las estampillas son impresas secuencialmente (orden cronológico) en filas, donde cada una de estas últimas corresponde a algún tipo de imagen o estructura como la imagen científica, estimación de flujo, etiquetas de píxeles, etc. (ver ejemplo de imagen en figura 4.3).

  Los valores `figsize1` y `figsize2` corresponden a la dimensión de la imagen resultante. El argumento `save_filename` indica el nombre con el que se quiere guardar el documento (de formato PNG) en disco.

- `print_space_states(obj, obs_rad, figsize1, figsize2, flux_thresh, rate_flux_thresh, save_filename)`:

  Esta función es la encargada de graficar la curva de estados por los que pasa un candidato (cuya información está contenida en el diccionario `obj`). Se visualizan los estados de tres épocas anteriores junto a los estados de las épocas en que ocurre su detección (mientras dure la alerta de detección). Estos estados están definidos por pares de valores de flujo y velocidad de flujo estimado por el filtro de Kalman. Además en la misma imagen, se indica el nivel de complejidad de la curva en términos de entropía [3]. Un ejemplo de imagen resultante se observa en la figura 4.4.

  Los argumentos del método, `figsize1` y `figsize2`, corresponden a las dimensiones de alto y ancho de la imagen a generar, mientras que `save_filename` indica el nombre del archivo generado a guardar.

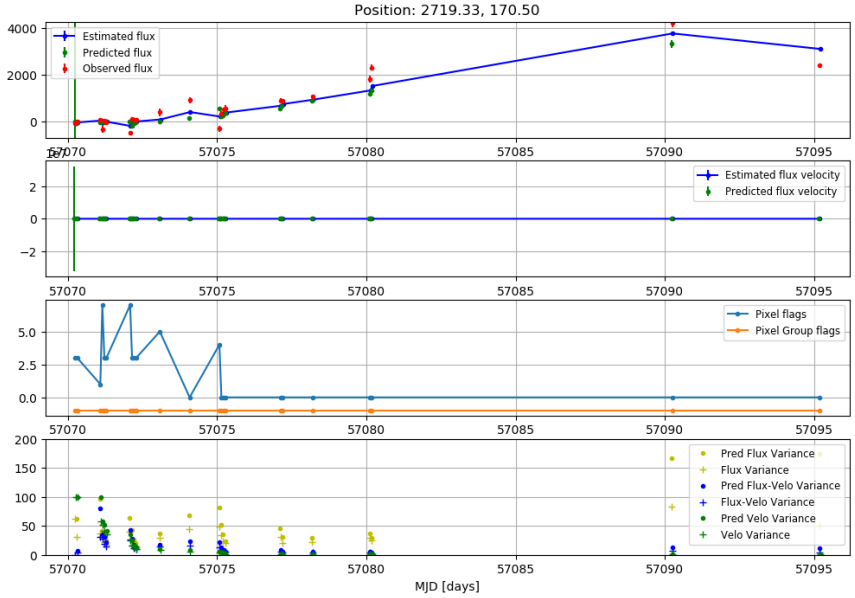


Figura 4.2: Conjunto de series de tiempo de diferentes componentes de interés en el pixel ubicado en la coordenada de posición del candidato. El primer gráfico (de arriba hacia abajo) muestra la evolución del flujo medido en contraste con el comportamiento del la predicción y estimación realizada por el filtro de Kalman del mismo flujo durante las épocas de las observaciones . La segunda visualización muestra los cambios de la predicción y estimación de la velocidad de flujo (obtenidas por el filtro de Kalman) en el tiempo. El tercer esquema muestra la evolución de las etiquetas (grupal e individual) del pixel ubicado en la coordenada del candidato. Finalmente, el último gráfico visualiza el comportamiento de las diferentes varianzas y covarianzas tanto de las componentes predichas y estimadas por el filtro (flujo y velocidad de flujo) así como de las mediciones del mismo flujo (observado).

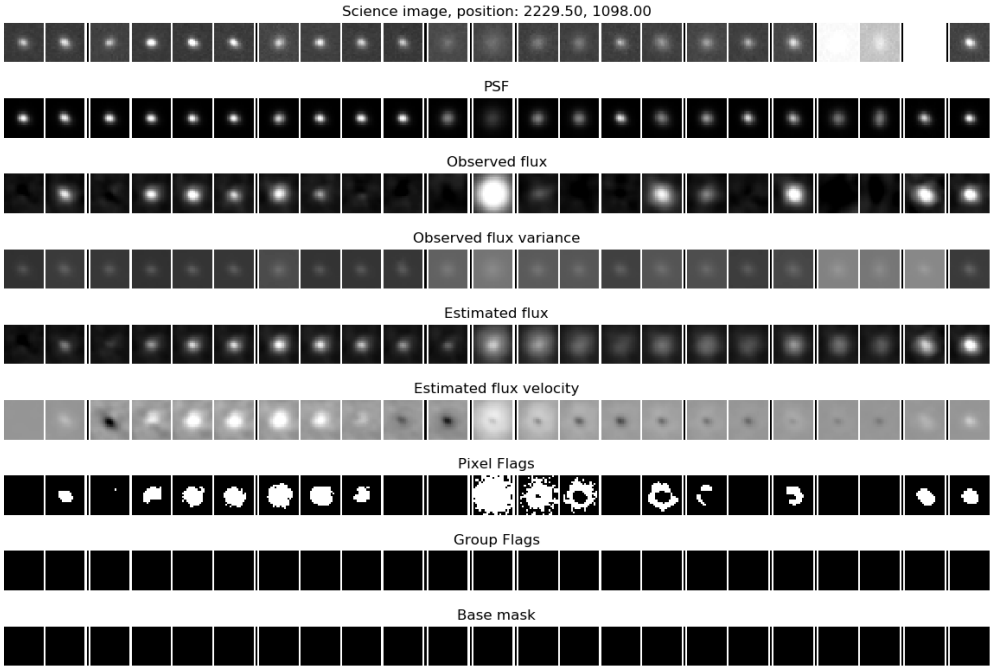


Figura 4.3: Estampillas de matrices de 21×21 píxeles y etiquetas que muestran el comportamiento de los píxeles y mediciones a través del tiempo: la primera fila de imágenes corresponde a estampillas obtenidas desde las imágenes científicas en donde debiese habitar la supernova observada durante todo el período de observación (definido por las épocas). La siguiente fila inferior muestra los diferentes modelos de PSF obtenidos para diferentes épocas. La tercera fila muestra el flujo observado en la misma posición. Le sigue la varianza de este flujo. Posteriormente viene el flujo estimado por el filtro de Kalman siguiendo la velocidad de flujo estimado. Finalmente vienen las etiquetas de los píxeles reconocidos por el programa como pertenecientes a un objeto transitorio (etiquetado por pixel y por grupo de píxeles). La última fila corresponde a la máscara base usada durante el análisis.

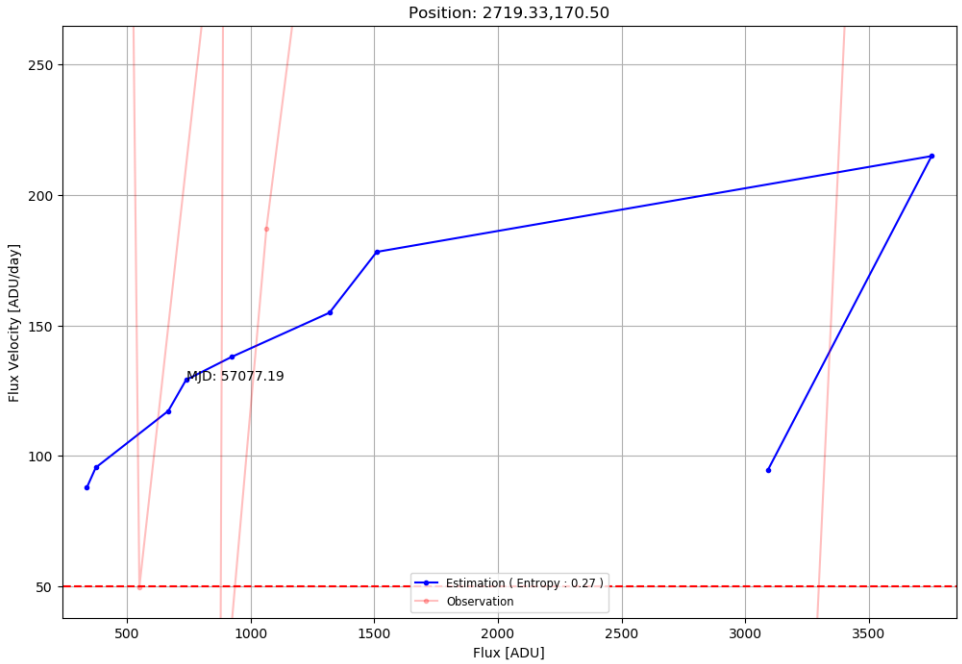


Figura 4.4: Espacio de fase de flujo y velocidad de flujo de un candidato. En azul se destaca la estimación lograda por filtro de Kalman. Notar que en la leyende de la figura, se indica el nivel de complejidad de la curva estimada en términos de su entropía [3], para la curva de flujo y velocidad de flujo estimado.

5 | Nueva funcionalidad

En este capítulo se detallan los cambios realizados a la manera de ejecutar la pipeline distinguiéndose de su antecesor RunData y la modularización de las diferentes tareas que involucra este programa, construyéndose así una nueva clase para la administración de los procesos: RoutineHandler. Además se describe la estructura y aspectos esenciales del nuevo miembro de la familia de filtros de Kalman: UnscentedKalman

5.1. Manejo de la rutina: RoutineHandler

La **rutina** comprende desde el proceso de listado de archivos (que en esta nueva versión se realiza con DataPicker), la creación de una instancia de la familia del filtro de Kalman y la secuencia iterativa de pasos de cálculo de flujo, obtención de estimaciones y el propio proceso de detección con SourceFinder. Para lograr esto, una instancia de RoutineHandler debe recibir tres archivos descritos a continuación:

- Lista de campos, CCDs, semestres y, alternativamente, las coordenadas de algún objeto de interés, en un archivo CSV (en este mismo orden) y con el encabezado `Field, CCD, Semester, POS_Y, POS_X`. En caso de no adjuntar las coordenadas en los campos de posicón, se puede rellenar con un guión ('-'). En el constructor este archivo se recibe como `obs_index_path`.
- Diccionario de directorios y expresiones regulares de las ubicaciones de los archivos y sus nombres, respectivamente (archivo de formato CSV). En el constructor de la clase este archivo se denomina como `route_templates`.
- Diccionario de umbrales y parámetros relevantes en la ejecución del programa, así como el tipo de filtro a usar (archivo de formato CSV). En el constructor se denominó `settings_file`.

Además de estos archivos se debe especificar el índice de la fila a procesar de la lista de campos (`index`), CCDs y semestre (primer archivo de entrada). Esto se hizo así con la finalidad de facilitar la paralelización de los análisis de diferentes conjuntos de datos. Por lo tanto el constructor de RoutineHandler queda como sigue:

`RoutineHandler(obs_index_path, route_templates, settings_file, index)`

5.1.1. Diccionario de umbrales y parámetros

El diccionario de umbrales y parámetros de configuración contiene la siguiente lista de valores:

- `imgHeight`: Altura de las imágenes científicas. Esta dimensión se propaga al resto de imágenes y matrices.
- `imgWidth`: Ancho de las imágenes científicas. Esta dimensión se propaga al resto de imágenes y matrices.
- `filter`: Tipo de filtro (`basic`, `mcc` o `ukf`).
- `results`: Directorio de resultados (donde guardar las coordenadas de los candidatos encontrados junto a lista de épocas en que fueron detectados) en formato NPZ.
- `init_var`: Varianza inicial que tendran las matrices de covarianza durante el proceso de estimación con los filtros de Kalman.
- `flux_thresh`: Umbral para estado de flujo obtenido con Kalman.
- `flux_rate_thresh`: Umbral para la velocidad de flujo obtenido con Kalman.
- `rate_satu`: Tasa de saturación.
- `sigma_a`: Varianza de la distribución de la componente de control (u_k) asumiendo normalidad. Es importante al emplear los filtros básico y unscented.
- `epsilon`: Radio de error con que la estimación por filtro de Kalman de máxima correntropía disminuye la ganancia de Kalman. Corresponde a un criterio de detención.
- `max_iter`: Número de iteraciones máximo para el proceso de corrección al usar Kalman de máxima correntropía.
- `silverman`: *Entero*. Toma valor 1 en caso de considerarse, y 0 si no. Se establece si se usa o no la aproximación de Silverman para determinar ancho de banda del kernel al emplear el filtro de máxima correntropía.
- `std_factor`: Factor de incremento de sigma al usar el método de Silverman.
- `sigma`: Sigma usado por defecto sin Silverman en la determinación del kernel durante el proceso de corrección con el Filtro de Kalman de correntropía máxima .

- `beta`: Parámetro relacionado con la distribución del estado estimado (x_k). Toma valor de $\beta = 2$ para distribuciones normales.

- `kappa`: Participa en la regulación del rango de los valores de los puntos sigma. Aporta incremento adicional (ver Ecuación 2.25).

- `alpha`: Participa en la regulación del rango de los valores de los puntos sigma. Incrementa el rango en un factor α (ver Ecuación 2.25).

- `dim`: Cantidad de componentes de estado a medir (en este programa se miden dos: flujo y su velocidad).

5.1.2. Métodos

Esta clase contiene los siguientes métodos:

- `process_settings()`:
  En este método se lee el archivo de diccionario de umbrales y parámetros con los que se configurará la toma de decisiones 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(check_found_objects)`:
  Recorre la lista de campos, CCDs y semestres entregada al programa con la consiguiente llamada a `routine`. Recibe como parámetro el argumento `check_found_objects` con el cual se indica si se quiere analizar resultados obtenidos anteriormente (candidatos encontrados), y que es entregado al método `routine` descrito a continuación.

- `routine(semester, field, ccd, results_path, check_found_objects, last_mjd)`:
  Comprende la rutina principal del programa, es decir, el análisis de las observaciones de un semestre, campo y CCD específico. El argumento `check_found_objects` es un boolean e indicará el modo de ejecución del método: si es falso, sólo guardará las coordenadas de los candidatos encontrados (si no encontró nada, entonces se guarda una lista vacía) además de

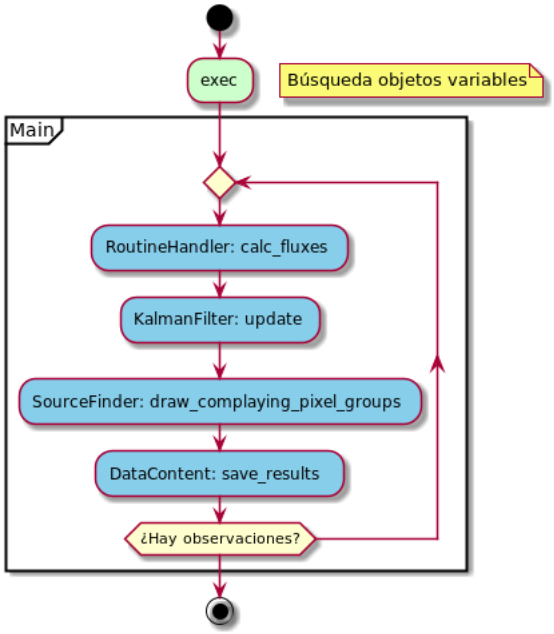


Figura 5.1: Rutina del programa refactorizado.

las épocas en que fueron detectados. Esta información se guarda en un arreglo de diccionarios. Si `check_found_objects` es verdadero, entonces cargará resultados anteriores del directorio de resultados (configurado en `process_settings`) para estudiar la presencia de los candidatos encontrados en caso de existir.

5.2. Filtro de Kalman Unscented

El diseño de la nueva clase de filtro se realizó continuando con la arquitectura del patrón Strategy, definiendo los métodos de predicción y corrección en instancias de clases que implementen las interfaces IPredict e ICorrect respectivamente: UnscentPredict y UnscentCorrect.

5.2.1. La clase UnscentedKalman

El filtro Unscented se desarrolló en la clase UnscentedKalman, la cual encierra los métodos de predicción y corrección (descritos en el Capítulo 2, subsección 2.3.3) implementados en `predict` y `correct`.

El constructor de esta clase requiere como argumentos de entrada los siguientes parámetros:

- `f_func`: Función usada para obtener el primer conjunto de puntos sigma para el período de predicción. Consiste en una función vectorial que se aplica sobre cada elemento de una matriz

cuya dimensión es la misma que la de la matriz de estado. Corresponde a la función principal con la que se aplicará el modelo no lineal en el proceso de predicción.

- `h_func`: Función usada para obtener el segundo conjunto de puntos sigma durante el proceso de corrección. Su aplicación procede de la misma forma que la descrita para `f_func`. Participa en el proceso de corrección.

- `f_args`: Corresponde a una lista de número reales de largo dos. El primer valor corresponde a la potencia o grado de la función mientras que el segundo término corresponde a el factor que la multiplica. Estos valores están asociados a la función `f`.

- `h_args`: Corresponde a una lista de número reales de largo dos dos. El primer valor corresponde a la potencia o grado de la función mientras que el segundo término corresponde a el factor que la multiplica. Estos valores están asociados a la función `h`.

- `alpha`: Término que indica que tan separados se encuentran los puntos sigma en torno a la media (junto a `kappa`). Por lo general su valor oscila entre 10^{-4} y 1.

- `beta`: Describe el valor de β el cual incorpora conocimiento *a priori* sobre la distribución de las variables de estado. Para distribuciones gaussianas, tiene un valor de $\beta = 2$.

- `kappa`: Corresponde a un parámetro de escalamiento secundario en la distancia de los puntos sigma a la media. Su valor va entre 0 y $3 - N$.

- `sigma_a`: Varianza asociada a la variable de control Δt. La función no lineal se aplicará sobre el paso del tiempo.

- `image_size`: Dimensiones de la imagen de flujo. Corresponde a una *tupla*.

Al inicializar una instancia de esta clase, se calculan tanto los pesos como el término λ, los cuales dependen de los parámetros: β, κ α y N (o cantidad de variables de estado). Ver Ecuaciones 2.24 y 2.25.

Como parte de la familia de KalmanFilter posee el método `update`, desde el cual se llama a `predict` y a `correct`, conservando la firma.

5.2.2. Predicción

La predicción de este filtro se implementó, como se mencionó anteriormente, en la clase UnscentedPredict (que implementa a IPredict). Para instanciar esta clase se debe entregar como

argumento al constructor: un puntero a función (que puede o no ser lineal) `f_func` y sus argumentos en `f_args`, los pesos asociados a la media `W_m` y la covarianza `W_c` de predicción, el coeficiente λ (`lambda_`), la cantidad de variables de estado medidas (en este modelo, se trata de dos: flujo y velocidad de flujo, por ende dos) `N`, la varianza σ_a (`sigma_a`) y la dimensión de las imágenes `image_size`. Por lo tanto la firma del constructor queda como sigue:

```
UnscentedPredict(f_func, f_args, Wm, Wc, lambda_, N, sigma_a, image_size)
```

Se destaca que tantos los pesos como el valor de λ se mantienen constantes durante todo el proceso de predicción y estimación, valores con los cuales se obtienen los puntos sigma y el valor medio al propagar la funcón `f_func` (ver Ecuación 2.22).

La salida de este método está compuesta por la matriz de estado predicha, la predicción de la covarianza y los puntos sigma guardados en la variable `Xs` de la implementación.

5.2.3. Corrección

El proceso de corrección está implementado en la clase UnscentedCorrect la cual implementa la interfaz ICorrect redefiniendo el método `correct` para esta versión del filtro. De acuerdo al proceso de corrección para el funcionamiento del filtro se deben definir las funciones f(ffl) y h(ffl) , además requiere de los pesos

El constructor de esta clase posee la siguiente firma.

```
UnscentedCorrect(f_func, h_func, f_args, h_args, Wm, Wc, lambda_, N, image_size)
```

Durante la corrección se escogen nuevamente $2N + 1$ puntos sigma en torno al estado predicho (fase previa) (ver Ecuación 2.28), sin embargo los valores de los pesos y de λ son los mismos usados durante la fase de predicción, cambiando sólo la función que se propaga sobre este nuevo conjunto de puntos: $h(\dot{})$ (expresada como `h_func`).

5.2.4. Funciones auxiliares

Se desarrollaron diferentes funciones auxiliares para apoyar el cálculo de las matrices:

- `sigma_points(mean_, cov_, lambda_, N):`
  Función con la cual se calculan los puntos sigma a partir la media de las variables de estado, la covarianza, el valor de λ y el número de variables de estado, N. Utiliza para esta finalidad, la descomposición de Cholesky.

- `unscent_weights(kappa, alpha, beta, N)`:

  Método con el que se calculan los pesos a partir de los valores de κ, α, β y N (número de variables de estado).

- `perform(func, *args)`:

  Función auxiliar con la cual se recibe un puntero a otra función vectorial (destinada a ser aplicada sobre un conjunto de puntos sigma) y un número arbitrario de argumentos, dependiendo de la necesidad de la misma función

- `propagate_func_pred(func, Wm, Wc, Xs, *args, N)`:

  Función con la que se propaga la función `func` sobre el conjunto de puntos sigma `Xs` usando los pesos de media y covarianza `Wm` y `Wc`, respectivamente, además del número de variables de estado, N. Además, recibe el argumento `args` que corresponde a una tupla de entradas propias de la función `func`.

- `propagate_func_corr`

Estas funciones están implementadas en el script `unscented_utils`.

6 | Resultados

En este capítulo se exponen resultados y análisis de diferentes pruebas realizadas con la nueva versión desarrollada del programa. La primera de las pruebas consiste en medir el desempeño de esta pipeline en términos de tiempo de ejecución y uso de memoria principal ocupada al procesar tres datasets (usados en el capítulo 3) asociados a supernovas encontradas por HiTS [8]: SN14, SN18 y SN80. El segundo conjunto de experimentos corresponde al procesamiento sobre los 93 conjuntos de datos asociados a las supernovas encontradas por HiTS en el semestre de 2015. En ambos tipos de pruebas se busca compara el rendimiento y resultados del programa al emplear los diferentes filtros de Kalman implementados. Finalmente se muestran algunos resultados específicos obtenidos con el filtro unscented.

Para todas las experiencias se empleó umbrales de flujo y velocidad de flujo estimada de 200 [ADU] y 50 [ADU/día]. Además todas las imágenes procesadas poseen la misma dimensión: 4094×2046 de alto por ancho, por lo que el campo `image_size` de todos los filtros se configuraron en (4094, 2046). Por cada filtro se emplearon los siguientes valores de parámetros:

Filtro de Kalman Básico

- `sigma_a` : Se ocupó un valor de 0.1.
- `init_var` : La varianza inicial de flujo y velocidad de flujo se estableció con un valor de 100.0.
- Se determinó 0.0 como valor inicial para las estimaciones de flujo y velocidad de flujo (estado).

Filtro de Kalman de máxima correntropía

- `sigma_a`: Se definió como 0.1
- `sigma`: Valor de varianza usado por defecto en caso de no usar optimización de Silverman.
- `silverman`: `False`. Para estas pruebas no se usó el método de Silverman para obtener distribución.

- `std_factor`: Se estableció como 100.0. Al no usarse el método de Silverman, tampoco se hace uso de este valor.
- `epsilon`: Se asignó valor de 10^{-6}.
- `max_iter`: Se consideró 10 como número máximo de iteraciones.
- `init_var` : La varianza inicial de flujo y velocidad de flujo se estableció con un valor de 100.0.
- Se determinó 0.0 como valor inicial para las estimaciones de flujo y velocidad de flujo (estado).

Filtro de Kalman Unscented

- `f_func:` Se definió una función no linear de grado 1.5 (información entregada en `f_args`).
- `h_func:` Se estableció como la función identidad.
- `f_args:` El grado de la función se definió como 1.5 y un factor de 1.0 (por tanto corresponde a la lista [1.5, 1.0]).
- `h_args:` Para estas pruebas se definió como una lista vacía ya que h(ffl) se trata de la función identidad.
- `alpha:` Se estableció como 0.001.
- `beta:` Se le asignó un valor de 2.0 (se asumió distribución normal de las variables de estado).
- `kappa:` Se le asignó un valor de 0.0 (se asumió distribución normal de las variables de estado).
- `N:` Al tratarse de dos variables de estado, se establece `N`=2.
- `sigma_a`: Se definió como 0.1.

Las condiciones iniciales del filtro unscented fueron diferentes a las definidas para los filtros básico y de máxima correntropía. Esto debido a que la función no lineal depende de un Δt medido desde cierta época t_0. Para estos experimentos se definió t_0 como la primera época de observación (los otros métodos dependen la diferencia temporal entre una época y otra). Por tanto, se configuró el flujo inicial al valor del flujo calculado de la primera época. Además, la varianza del estado del flujo fue definida como la matriz de varianza de flujo calculado.

6.1. Desempeño

En esta sección se describen los resultados obtenidos durante las experiencias de medición de desempeño de la nueva versión del programa en términos de tiempo y memoria principal usada.

Las pruebas se realizaron usando el mismo conjunto de datos utilizado para medir el desempeño del programa original (Capítulo 3), además se emplearon como herramientas de medición igualmente el perfilador de memoria `mem_profiler` y la librería RESOURCE[1].

6.1.1. Tiempo de ejecución

Las Tablas 6.1, 6.2 y 6.3 muestran los resultados de tiempo medido en segundos de cada una de las partes del proceso de la rutina principal, para los tres datasets usando los filtros básico, de máxima correntropía y unscented, respectivamente.

Tabla 6.1: Tiempo de ejecución en segundos de cada proceso involucrado, usando el filtro de Kalman básico refactorizado: cálculo de flujo, estimación de estados, detección de candidatos y guardado de candidatos resultantes en caso de haberlos. La última fila describe el tiempo promedio que toma por observación (en segundos igualmente) para cada uno de los procesos.

ID	**Cálc. Flujos [s]**	**Estim. KF [s]**	**Detección [s]**	**Obt. Candidatos [s]**
SN14	297.93	28.36	33.53	0.00
SN18	255.07	22.41	34.32	0.00
SN80	199.16	17.73	25.02	0.00
$\bar{t}/Obs$	11.20	1.02	1.39	0.00

Tabla 6.2: Tiempo de ejecución en segundos de cada proceso involucrado, usando el filtro de Kalman de máxima correntropía refactorizado: cálculo de flujo de las imágenes, estimación de estado, detección de fuentes y guardado de candidatos. La última fila describe el tiempo promedio que toma por observación (en segundos) para cada una de las tareas.

ID	**Cálc. Flujos [s]**	**Estim. KF [s]**	**Detección [s]**	**Actual. Candidatos [s]**
SN14	289.59	491.08	34.04	0.00
SN18	256.95	437.37	33.00	0.00
SN80	199.98	346.98	24.96	0.00
$\bar{t}/Obs$	11.00	19.06	1.38	0.00

A todos estos resultados se les ha agregado la medición de tiempo promedio por imagen (ya que las secuencias poseen largos diferentes). De estos resultados es posible deducir que el tiempo de cálculo de flujo por época siempre toma alrededor de 11.0 segundos. Esto es claro debido a que el proceso de cálculo de flujo es independiente del filtro escogido. Los tiempos de estimación de estado, por el contrario, varían de filtro en filtro. Tomando mayor tiempo al usar el filtro de máxima

[1] Método `get_rusage`

Tabla 6.3: Tiempo de ejecución en segundos de cada tarea, usando el filtro de Kalman unscented: cálculo de flujo, estimación de filtros, detección de fuentes y guardado de candidatos. La última fila describe el tiempo promedio que toma por observación (en segundos) para cada uno de los procesos.

ID	**Cálc. Flujos [s]**	**Estim. KF [s]**	**Detección [s]**	**Actual. Candidatos [s]**
SN14	280.67	342.88	123.68	0.01
SN18	243.96	293.05	122.09	0.00
SN80	190.43	231.34	74.48	0.00
$\bar{t}/Obs$	10.65	12.93	14.21	0.00

correntropía. Sin embargo, opuesto a lo que se esperaba (ya que el proceso de detección ocurre una vez realizada la estimación de estado) el filtro unscented presenta un tiempo de detección casi 10 veces mayor a los tiempos medidos en los otros filtros.

Con los filtros básico y de máxima correntropía no se encontraron candidatos en ninguno de los datasets. Sin embargo se detectaron candidatos con el filtro unscented al ejecutar la pipeline sobre los datos de la supernova SN14.

La totalidad y el promedio del tiempo que toma la ejecución de la pipeline con los tres filtros sobre los datasets se muestran en las tablas 6.4, 6.5 y 6.6 (filtros básico, de correntropía máxima y unscented, respectivamente).

Tabla 6.4: Tiempo de exploración (para detección de candidatos) usando filtro de Kalman Básico refactorizado. La última fila corresponde a tiempo total promedio por observación.

ID	**Tiempo total [s]**
SN14	359.82
SN18	311.8
SN80	241.91
$\bar{t}/Obs.$	13.61

Tabla 6.5: Tiempo de exploración (para detección de candidatos) usando filtro de Kalman de máxima correntropía refactorizado. La última fila corresponde a tiempo total promedio por observación.

ID	**Tiempo total [s]**
SN14	814.71
SN18	727.32
SN80	571.92
$\bar{t}/Obs.$	31.58

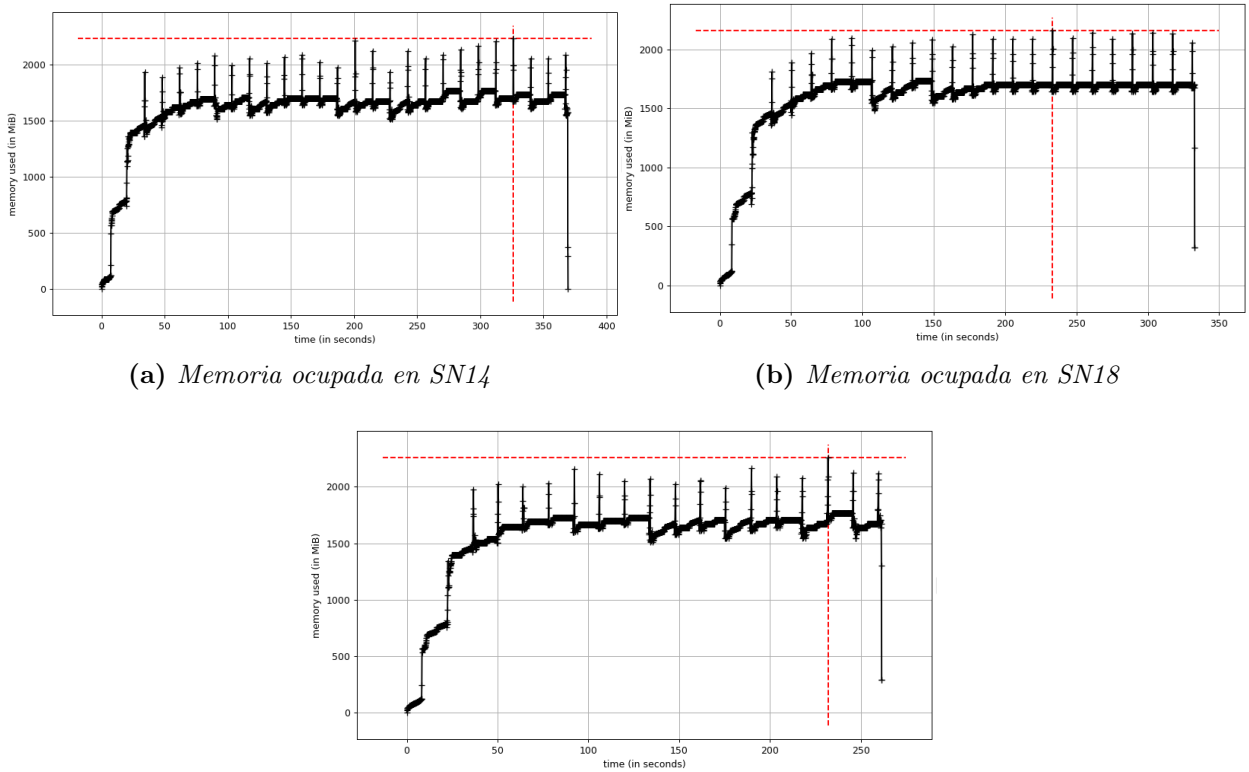
De los resultados expuestos sobre el tiempo total de ejecución se desprende que la pipeline demora menos con el filtro básico, mientras que con el de máxima correntropía es con el que más se demora.

Tabla 6.6: Tiempo de exploración (para detección de candidatos) usando filtro unscented. La última fila corresponde a tiempo total promedio por observación en segundos.

ID	**Tiempo total [s]**
SN14	747.24
SN18	659.1
SN80	496.25
$\bar{t}/Obs.$	28.32

6.1.2. Uso de memoria

Las figuras 6.1, 6.2 y 6.3 muestran el comportamiento del consumo de memoria al usar la nueva versión del filtro básico, de máxima correntropía y el nuevo filtro unscented, respectivamente, ejecutando la pipeline refactorizada sobre los tres conjuntos de datos (SN14, SN18 y SN80).



(a) *Memoria ocupada en SN14*

(b) *Memoria ocupada en SN18*

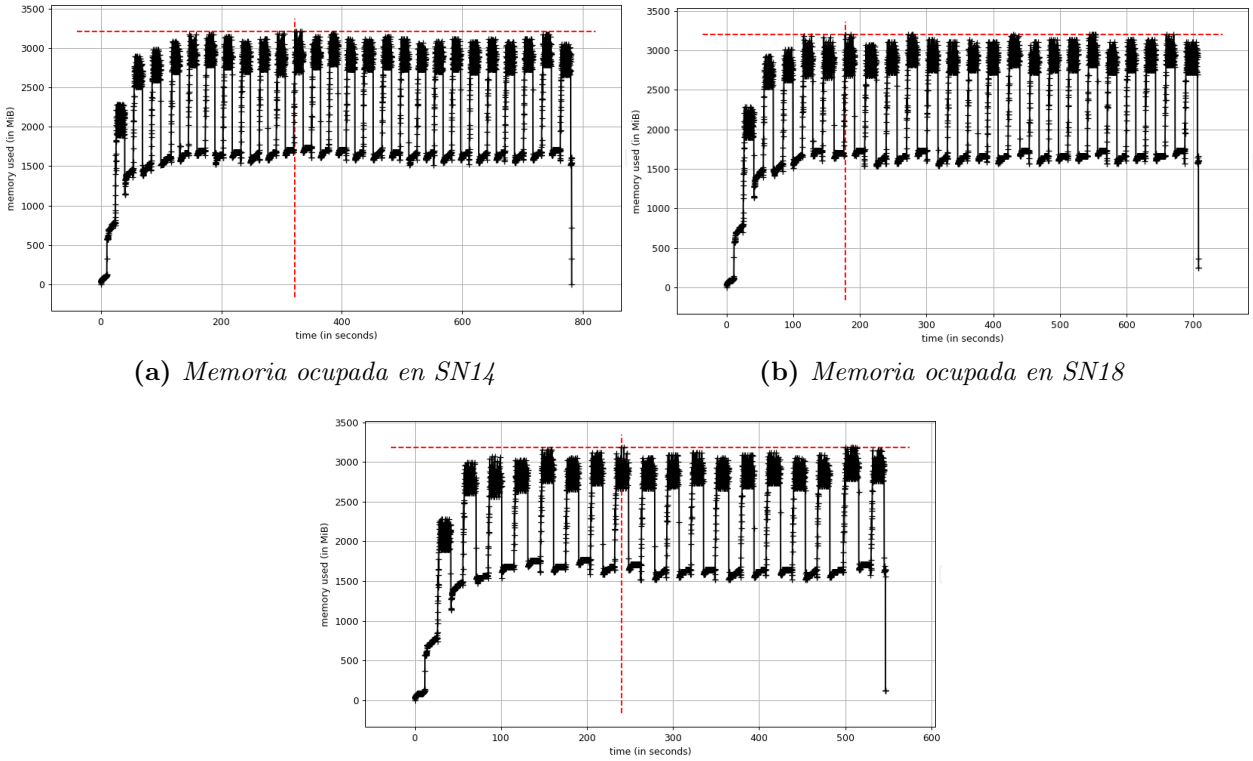
(c) *Memoria ocupada en SN80*

Figura 6.1: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos usando el filtro de Kalman Básico.

De la Figura 6.1 se desprenden los máximos alcanzados durante cada ejecución con el filtro básico. Los resultados en megabytes se listan en la tabla 6.7.

Tabla 6.7: Memoria principal (en unidades de MB) usada durante la ejecución del programa refctorizado usando filtro de Kalman Básico.

ID	Memoria [MB]
SN14	2347.78
SN18	2254.93
SN80	2365.07



(a) *Memoria ocupada en SN14*

(b) *Memoria ocupada en SN18*

(c) *Memoria ocupada en SN80*

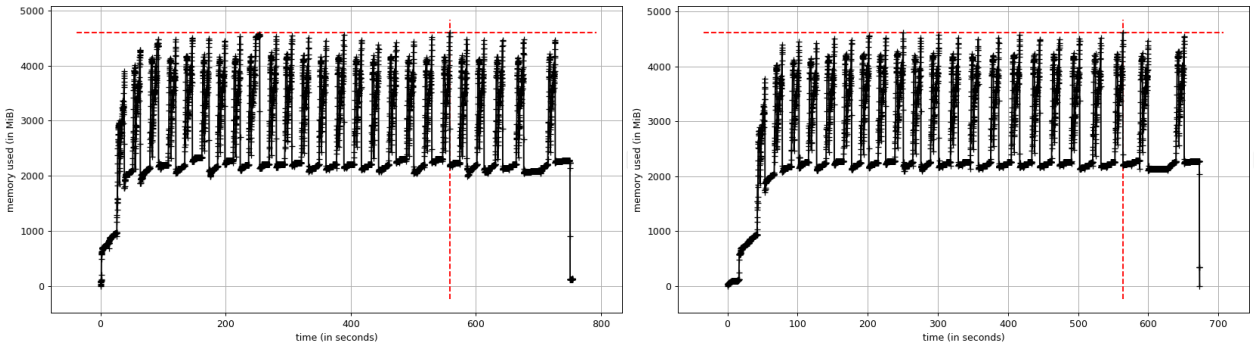
Figura 6.2: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman de máxima correntropía.

De las series mostradas en la figura 6.2 se obtienen los máximos alcanzados de memoria principal ocupada al utilizar el filtro de máxima correntropía listados en la Tabla 6.8.

Tabla 6.8: Memoria principal (en unidades de MB) usada durante la ejecución del programa refactorizado usando filtro de Kalman de máxima correntropía.

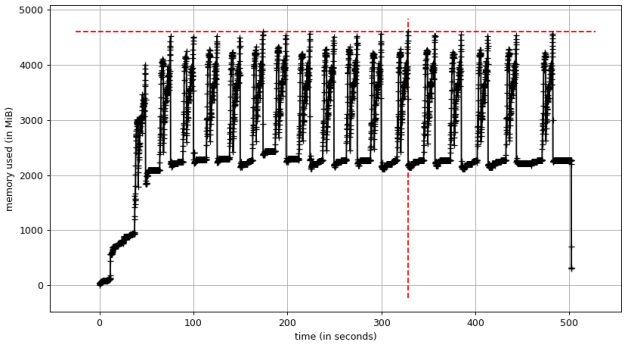
ID	Memoria [MB]
SN14	3359.54
SN18	3350.53
SN80	3330.93

Los máximos de uso de memoria principal al emplear el filtro unscented, para cada conjunto de datos, se muestra en la Tabla 6.9.



(a) *Memoria ocupada en SN14*

(b) *Memoria ocupada en SN18*



(c) *Memoria ocupada en SN80*

Figura 6.3: 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 unscented.

Tabla 6.9: Memoria principal (en unidades de MB) usada durante la ejecución del programa refactorizado usando filtro de Kalman unscented.

ID	Memoria [MB]
SN14	4827.82
SN18	4835.96
SN80	4821.89

Los resultados de memoria principal muestran un uso intensivo de esta al usar el filtro unscented. Esto puede ser debido a la serie de operaciones lineales que se debe realizar al momento de predecir y corregir. También se desprende que el uso de memoria es menor con el filtro básico.

6.2. Detección

En esta sección se detallan los resultados obtenidos para esta nueva pipeline sobre los datos en las 93 supernovas conocidas. Para todos los filtros listados se empleó un threshold de flujo de 200 y de velocidad de flujo de 50.

6.2.1. Filtros básico y de máxima correntropía

La Tabla 6.10 muestra los resultados obtenidos con la nueva versión de la pipeline, empleando los filtros refactorizados de la familia strategy.

Tabla 6.10: Número de falsos negativos (FN) y verdaderos positivos (TP) encontrados usando cada uno de los filtros. No se observan diferencias entre los resultados de cada filtro refactorizado. La tercera columna de valores muestra la cantidad de conjuntos de datos que no pudieron ser procesados.

Filtro	**TP**	**FN**	**NaN**
Básico	37	56	3
MCC	37	56	3

Los resultados muestran que para el programa refactorizado tanto el filtro básico como el de correntropía máxima detectan la misma cantidad de supernovas de HiTS. De acuerdo a la información dispuesta en la sección F apéndice, donde se muestra la tabla 7.2, se desprende que tampoco hay diferencias de los tiempos o épocas de detección.

Al comparar las tablas 7.1, correspondiente a los tiempos de detección en MJD de las implementaciones originales de los filtros y 7.2 a los tiempos de las versiones refactorizadas, se deduce que las detecciones con el último grupo de filtros ocurren, cuando es el caso, en la misma época o antes (horas o días). Por ejemplo, en las imágenes 3.7 y 3.8 las detecciones se realizan en 57072.145 y 57075.12 con el nuevo conjunto de filtros. En otras ocasiones, alguno de los grupos no detecta la supernova y en el peor de los casos ninguno de los dos reconoce la supernova.

6.2.2. Filtro unscented

Una vez que se pasen las pruebas (`unittests`) de correctitud, se procederá a ejecutar las pruebas del filtro Unscented en Leftraru.

7 | Conclusiones

Se propone una nueva versión optimizada del software implementado originalmente por Pablo Huentelemu, obteniéndose una familia de filtros de Kalman que puede ser fácilmente extendida con la finalidad de agregar nuevos métodos de estimación de estado. Esta extensión es facilitada gracias al patrón de diseño Strategy.

Por otro lado, otro de los beneficios adquiridos en esta versión actual de programa es el control sobre los argumentos de entrada al programa. Anteriormente, los nombres de los archivos tenían que ser escritos explícitamente en el código del programa. Sin embargo el código rediseñado en este trabajo es independiente de los nombres de los archivos que recibe como entrada por lo que el usuario sólo debe velar en entregar las rutas y las expresiones regulares de las imágenes correctas en los archivos de entrada de esta nueva versión del programa. Además, cabe destacar que igualmente se tiene se independizó del código los valores de entrada de umbrales y de configuración requeridos en diferentes rutinas entregándose estos en un archivo de texto plano (en donde se incluye el campo y su valor). Es decir se logró evitar el *harcoding* presente en la versión anterior de la pipeline.

Respecto de las pruebas realizadas en la evaluación de las versiones original y nueva, se observa que efectivamente la nueva versión de los filtros básico y de máxima correntropía logran detectar más supernovas que sus respectivas versiones originales, usando un umbral de flujo de 200 y de 50 en velocidad de flujo. Este resultado se presume que puede ser debido a la mejora en el ordenamiento de los datos de entrada ya que este ordenamiento es aplicado sobre todos los datasets de entrada y no sólo sobre las imágenes científicas como lo realiza el programa original.

Se desprende del análsis de desempeño que, al menos para el par de valores de umbral de flujo y velocidad de flujo entregadas no hay diferencia entre los resultados entregados por un filtro u otro (para cada versión, orginal o nueva), respecto de los filtros de Kalman básico y máxima correntropía, empleando un modelo linear. Por lo que, al escoger un modelo linear para el estudio de candidatos en algún nuevo conjunto de datos, se recomienda utilizar el filtro Básico debido a su bajo consumo de memoria y tiempo. Esto, ya que aparentemente no hay diferencias con el filtro de

máxima correntropía en sus estimaciones. Sin embargo, se debe hacer notar que estos resultados también dependen de las condiciones iniciales y parámetros que reciba cada filtro de acuerdo a su arquitectura. El conjunto de parámetros seleccionado para ambos filtros durante la generación de este trabajo permitió la obtención de resultados similares.

Se debe agregar, que dentro de los cambios agregados a la nueva versión de la pipeline se acortó tiempo del proceso ya que a priori se asume que no se conoce ninguna supernova, es decir, el programa no tiene porqué saber si existe un candidato conocido, por lo que los resultados de los potenciales aspirantes a supernova son tratados ecuánimamente, y por tanto no se discrimina en la información obtenida en los resultados al ser guardada en disco. Por ende, el análisis se realiza en una pasada (lo que le permite demorarse la mitad del tiempo original) y no en dos como se realizaba antigüamente.

7.1. Trabajo futuro

Los métodos presentados pueden ser aplicados en la detección de estrellas variables aplicando un filtro que distinga tendencias decrecientes en la luminosidad de estos objetos; por lo que una extensión prometedora de este sistema podría diseñarse para reconocer alternancia en los regímenes creciente y decreciente, haciendo uso de la estructura de clases dada por el patrón Strategy en el modelo del filtro.

Por otra parte queda también pendiente, el estudiar el comportamiento de los resultados al variar los umbrales relacionados con la estimación realizada por el filtro de Kalman y que son usados en la fase de reconocimiento de candidatos (en la clase SourceFinder).

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. [14]. Ver imagen 7.1

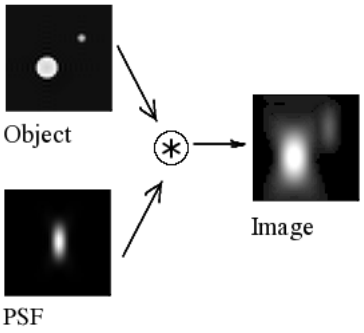


Figura 7.1: Ejemplo de distorsión de una fuente al aplicar un kernel de PSF específico. El resultado se observa en el cuado *Image*.

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/apps/astro/data/DATA/Blind%s_%s/%s
scienceDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
diffDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
invDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
afluxDir = /home/apps/astro/data/SHARED/Blind%s_%s/%s/CALIBRATIONS
psfDir = /home/apps/astro/data/SHARED/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.

E . Detección usando pipeline original

Tabla 7.1: Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando los filtros implementados originalmente (básico y de correntropía máxima).

Índ.	Básico	MCC
1	57072.1869637	57072.1869637
2	-	-
3	57075.1450198	57075.1450198
4	57075.2134724	57075.2134724
5	57072.2392548	57072.2392548
6	-	-
7	-	-
8	57075.1009618	57075.1009618
9	57075.2202993	57075.2202993
10	57077.1097248	57077.1097248
11	57075.2038295	57075.2038295
12	57072.2143034	57072.2143034
13	57077.1546506	57077.1546506
14	-	-
15	57077.1110959	57077.1110959
16	57077.0857227	57077.0857227
17	57077.1110959	57077.1110959
18	57090.2264287	57090.2264287
19	-	-
20	57077.1178	57077.1178
21	-	-
22	57077.1230908	57077.1230908
23	-	-
24	57077.08304	57077.08304
25	57075.2452892	57075.2452892
26	-	-
27	57077.1656386	57077.1656386
28	-	-
29	-	-
30	57072.3480749	57072.3480749
31	-	-
32	57075.2106837	57075.2106837
33	-	-
34	-	-
35	-	-
36	57080.1051523	57080.1051523
37	57075.2134724	57075.2134724
38	-	-
39	-	-
40	-	-
41	-	-
42	-	-
43	57090.2200634	57090.2200634
44	-	-
45	-	-
46	-	-
47	57075.1957933	57075.1957933
48	57080.1745874	57080.1745874
49	57090.2390433	57090.2390433
50	57075.2051724	57075.2051724
51	-	-
52	-	-
53	-	-
54	-	-
55	-	-
56	57075.107637	57075.107637
57	57095.1962245	57095.1962245
58	-	-
59	57095.2015731	57095.2015731
60	-	-
61	-	-
62	57095.1619457	57095.1619457
63	-	-
64	-	-
65	-	-
66	-	-
67	-	-
68	-	-
69	-	-
70	-	-
71	-	-
72	-	-
73	-	-
74	-	-
75	-	-
76	-	-
77	-	-
78	-	-
79	-	-
80	-	-
81	-	-
82	-	-
83	-	-
84	-	-
85	-	-
86	57080.2017908	57080.2017908
87	-	-
88	-	-
89	-	-
90	-	-
91	NaN	NaN
92	NaN	NaN
93	NaN	NaN

F . Detección usando pipeline refactorizada

Tabla 7.2: Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando los filtros refactorizados (básico y de correntropía máxima).

Índ.	Básico	MCC
1	57072.186964	57072.186964
2	-	-
3	57075.14502	57075.14502
4	57072.152555	57072.152555
5	57072.239255	57072.239255
6	57077.085723	57077.085723
7	-	-
8	57075.033352	57075.033352
9	57075.220299	57075.220299
10	57077.109725	57077.109725
11	57075.20383	57075.20383
12	57072.144542	57072.144542
13	57077.154651	57077.154651
14	57077.19278	57077.19278
15	57077.111096	57077.111096
16	57075.168335	57075.168335
17	57077.111096	57077.111096
18	57080.097143	57080.097143
19	-	-
20	57077.179284	57077.179284
21	57077.123091	57077.123091
22	57077.123091	57077.123091
23	-	-
24	-	-
25	57075.245289	57075.245289
26	57077.087046	57077.087046
27	57075.119637	57075.119637
28	-	-
29	-	-
30	57072.279068	57072.279068
31	-	-
32	57075.210684	57075.210684
33	-	-
34	-	-
35	57077.194101	57077.194101
36	57080.105152	57080.105152
37	-	-
38	(57077.200762)	(57077.200762)
39	-	-
40	-	-
41	-	-
42	-	-
43	57090.220063	57090.220063
44	-	-
45	-	-
46	-	-
47	57072.204787	57072.204787

Índ.	Básico	MCC
48	57080.106472	57080.106472
49	57090.239043	57090.239043
50	57077.121768	57077.121768
51	57077.132439	57077.132439
52	-	-
53	-	-
54	-	-
55	-	-
56	57075.107637	57075.107637
57	57090.252927	57090.252927
58	-	-
59	-	-
60	-	-
61	-	-
62	57095.161946	57095.161946
63	-	-
64	-	-
65	-	-
66	57095.146651	57095.146651
67	-	-
68	-	-
69	-	-
70	-	-
71	-	-
72	-	-
73	-	-
74	-	-
75	-	-
76	-	-
77	-	-
78	-	-
79	-	-
80	-	-
81	-	-
82	-	-
83	-	-
84	-	-
85	-	-
86	57080.201791	57080.201791
87	-	-
88	-	-
89	-	-
90	-	-
91	NaN	NaN
92	NaN	NaN
93	NaN	NaN

Referencias

[1] H. Aihara, N. Arimoto, R. Armstrong, S. Arnouts, N. A. Bahcall, S. Bickerton, J. Bosch, K. Bundy, P. L. Capak, J. H. Chan, et al. The hyper suprime-cam ssp survey: overview and survey design. *Publications of the Astronomical Society of Japan*, 70(SP1):S4, 2017.

[2] M. Arnaboldi, M. Capaccioli, D. Mancini, R. Scaramella, G. Sedmak, and R. Kurz. The vst-vlt survey telescope. In *From Extrasolar Planets to Cosmology: The VLT Opening Symposium*, pages 204–208. Springer, 2000.

[3] A. Balestrino, A. Caiti, and E. Crisostomi. Generalised entropy of curves for the analysis and classification of dynamical systems. *Entropy*, 11(2):249–270, 2009.

[4] R. D. Blum, K. Burleigh, A. Dey, D. J. Schlegel, A. M. Meisner, M. Levi, A. D. Myers, D. Lang, J. Moustakas, A. Patej, et al. The decam legacy survey. In *American Astronomical Society Meeting Abstracts# 228*, volume 228, 2016.

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

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

[7] J. Emerson, W. Sutherland, A. McPherson, S. Craig, G. Dalton, and A. Ward. The visible & infrared survey telescope for astronomy. *The Messenger*, 117:27–32, 2004.

[8] 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.

[9] 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.

[10] N. Kaiser, H. Aussel, B. E. Burke, H. Boesgaard, K. Chambers, M. R. Chun, J. N. Heasley, K.-W. Hodapp, B. Hunt, R. Jedicke, et al. Pan-starrs: a large synoptic survey telescope array. In *Survey and Other Telescope Technologies and Discoveries*, volume 4836, pages 154–165. International Society for Optics and Photonics, 2002.

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

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

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

[14] 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.

[15] 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.