

# Filtros de ruido speckle y técnicas de clasificación para Imágenes SAR implementadas en Python

Herranz, Matías    Tita, Joaquín

Facultad de Matemática, Astronomía y Física  
Universidad Nacional de Córdoba

15 de septiembre de 2012

# Contenidos

- 1 Introducción
- 2 Definiciones y Contexto
- 3 Definiciones algorítmicas y referencias matemáticas
- 4 Implementación Computacional
- 5 Resultados, Ejemplos y Performance

# Motivaciones y disparadores

- Primer contacto con el procesamiento de imágenes,

# Motivaciones y disparadores

- Primer contacto con el procesamiento de imágenes,
- Ausencia de alternativas libres,

# Motivaciones y disparadores

- Primer contacto con el procesamiento de imágenes,
- Ausencia de alternativas libres,
- Proyecto de tesis con retribución a la sociedad,

# Motivaciones y disparadores

- Primer contacto con el procesamiento de imágenes,
- Ausencia de alternativas libres,
- Proyecto de tesis con retribución a la sociedad,
- Apertura y continuidad.

# Repercusión social

- Comparando resultados: *ENVI* y problemas que buscábamos tener,

# Repercusión social

- Comparando resultados: *ENVI* y problemas que buscábamos tener,
- Brindar una opción distinta,

# Repercusión social

- Comparando resultados: *ENVI* y problemas que buscábamos tener,
- Brindar una opción distinta,
- Poder de la comunidad,

# Repercusión social

- Comparando resultados: *ENVI* y problemas que buscábamos tener,
- Brindar una opción distinta,
- Poder de la comunidad,
- Libertad y accesibilidad.

# Por qué Python

- Python
  - ▶ es software libre,

# Por qué Python

- Python

- ▶ es software libre,
- ▶ tiene una comunidad muy activa y cooperativa,

# Por qué Python

- Python

- ▶ es software libre,
- ▶ tiene una comunidad muy activa y cooperativa,
- ▶ es un lenguaje con una sintaxis sencilla y es fácil de entender y aprender,

# Por qué Python

- Python

- ▶ es software libre,
- ▶ tiene una comunidad muy activa y cooperativa,
- ▶ es un lenguaje con una sintaxis sencilla y es fácil de entender y aprender,
- ▶ es rápido,

# Por qué Python

- Python

- ▶ es software libre,
- ▶ tiene una comunidad muy activa y cooperativa,
- ▶ es un lenguaje con una sintaxis sencilla y es fácil de entender y aprender,
- ▶ es rápido,
- ▶ tiene un gran soporte para computación científica,

# Por qué Python

- Python

- ▶ es software libre,
- ▶ tiene una comunidad muy activa y cooperativa,
- ▶ es un lenguaje con una sintaxis sencilla y es fácil de entender y aprender,
- ▶ es rápido,
- ▶ tiene un gran soporte para computación científica,
- ▶ es multiplataforma.

# Filtros de ruido speckle

- ¿Qué es el ruido speckle?

# Filtros de ruido speckle

- ¿Qué es el ruido speckle?
- ¿Cómo se forma?

# Filtros de ruido speckle

- ¿Qué es el ruido speckle?
- ¿Cómo se forma?
- ¿Cómo se manifiesta visualmente?

# Filtros de ruido speckle

- ¿Qué es el ruido speckle?
- ¿Cómo se forma?
- ¿Cómo se manifiesta visualmente?



Imagen de Radar con ruido speckle de la zona de Santa Rosa de Río Primero gentileza de CONAE.

# Modelo Matemático utilizado

$$I(t) = R(t) \cdot u(t)$$

Donde:

- $t = (x, y)$ ,
- $I(t)$ ,
- $R(t)$ ,
- $u(t)$ .

# Filtro de Kuan

$$\hat{R}(t) = I(t) \cdot W(t) + \bar{I}(t) \cdot (1 - W(t))$$

Donde la función de peso es:

$$W(t) = \frac{1 - C_u^2/C_I^2(t)}{1 + C_u^2}$$

con:

- $C_u = \sigma_u/\bar{u}$ ,
- $C_I = \sigma_I/\bar{I}$ ,
- $I(t)$ ,
- $\bar{I}(t)$ .

# Filtro de Lee

$$\hat{R}(t) = I(t) \cdot W(t) + \bar{I}(t) \cdot (1 - W(t))$$

Donde la función de peso es:

$$W(t) = 1 - C_u^2 / C_I^2(t)$$

con:

- $C_u = \frac{\sigma_u}{\bar{u}}$ ,
- $C_I = \frac{\sigma_I}{I}$ ,
- $I(t)$ ,
- $\bar{I}(t)$ ,
- Limitación del filtro de Lee cuando  $C_u \gg C_I$ .

# Filtro de Lee Mejorado (Lee Enhanced)

$$\hat{R}(t_0) = \begin{cases} \bar{I}(t_0), & \text{si } C_I(t_0) \leq C_u \\ I(t_0) \cdot W(t_0) + \bar{I}(t_0) \cdot (1 - W(t_0)), & \text{si } C_u \leq C_I(t_0) < C_{max} \\ I(t_0), & \text{si } C_I(t_0) \geq C_{cmax} \end{cases}$$

Donde:

$$W(t_0) = \exp[-K(C_I(t_0) - C_u)/(C_{max} - C_I(t_0))]$$

- $K$ , es el factor de amortiguamiento,
- $C_{max}$ , es el máximo coeficiente de variación del ruido.

# Filtro de Frost

$$m(t) = e^{-KC_l(t_0)|t|}$$

con:

- $K$ ,
- $t_0$ ,
- $C_l(t_0)$ .

# Introducción a Clasificación de Imágenes

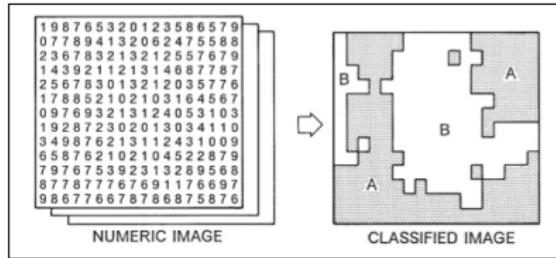
- En qué consiste la clasificación,

# Introducción a Clasificación de Imágenes

- En qué consiste la clasificación,
- Ejemplos/aplicaciones,
  - ▶ Aprendizaje automático,
  - ▶ Reconocimiento de patrones,
  - ▶ Bioinformática.

# Introducción a Clasificación de Imágenes

- En qué consiste la clasificación,
- Ejemplos/aplicaciones,
  - ▶ Aprendizaje automático,
  - ▶ Reconocimiento de patrones,
  - ▶ Bioinformática.
- Ejemplo visual:



# Clasificación No Supervisada

- Concepto general,

# Clasificación No Supervisada

- Concepto general,
- K-Means.

# Clasificación Supervisada

- Concepto general,

# Clasificación Supervisada

- Concepto general,
- Isodata.

# PyRadar



# Estructura de Archivos y Carpetas de PyRadar

```
pyradar/
├── classifiers
├── comparator
├── core
├── examples
├── filters
├── simulate
└── utils
```

# Módulo *classifiers*

```
pyradar/
└── classifiers
    ├── isodata.py
    └── kmeans.py
```

# Módulo *comparator*

```
pyradar/
└── comparator
    ├── comparator_utils.py
    └── image_comparator.py
```

# Módulo core

```
pyradar/
└── core
    ├── equalizers.py
    └── sar.py
```

# Módulo *filters*

```
pyradar/
└── filters
    ├── frost.py
    ├── kuan.py
    ├── lee.py
    ├── lee_enhanced.py
    ├── mean.py
    ├── median.py
    └── utils.py
```

# Módulo *simulate*

```
pyradar/
    └── simulate
        ├── ExampleImages
        │   ├── DemoSet
        │   │   ├── gamma_img_layer.png
        │   │   ├── gamma_noisy_img.png
        │   │   ├── k_img_layer.png
        │   │   ├── k_noisy_img.png
        │   │   ├── demo_K.png
        │   │   └── demo_gamma.png
        └── image_simulator.py
```

# Módulos *utils* y *examples*

```
pyradar/
└── utils
    ├── sar_debugger.py
    ├── statutils.py
    ├── system_info.py
    └── timeutils.py
└── examples
    └── ...
```

# Performance

- ENVI vs. PyRadar,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,
- Profiling,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,
- Profiling,
- Uso de librerías externas,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,
- Profiling,
- Uso de librerías externas,
- Posibilidades de nuevas mejoras,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,
- Profiling,
- Uso de librerías externas,
- Posibilidades de nuevas mejoras,
  - ▶ Threading,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,
- Profiling,
- Uso de librerías externas,
- Posibilidades de nuevas mejoras,
  - ▶ Threading,
  - ▶ GPGPUs,

# Performance

- ENVI vs. PyRadar,
- Cómo manejamos/implementamos las mejoras de performance,
  - ▶ Implementación progresiva,
  - ▶ Identificación de partes críticas,
- Profiling,
- Uso de librerías externas,
- Posibilidades de nuevas mejoras,
  - ▶ Threading,
  - ▶ GPGPUs,
  - ▶ Uso de lenguajes de bajo nivel para algunas porciones críticas de código.

# Resultados y Ejemplos

- Kuan

# Resultados y Ejemplos

- Kuan
- Isodata

# Resultados y Ejemplos

- Kuan
- Isodata
- Frost + K-Means

# Resultados y Ejemplos

- Kuan
- Isodata
- Frost + K-Means
- Video de ejemplo de Isodata.

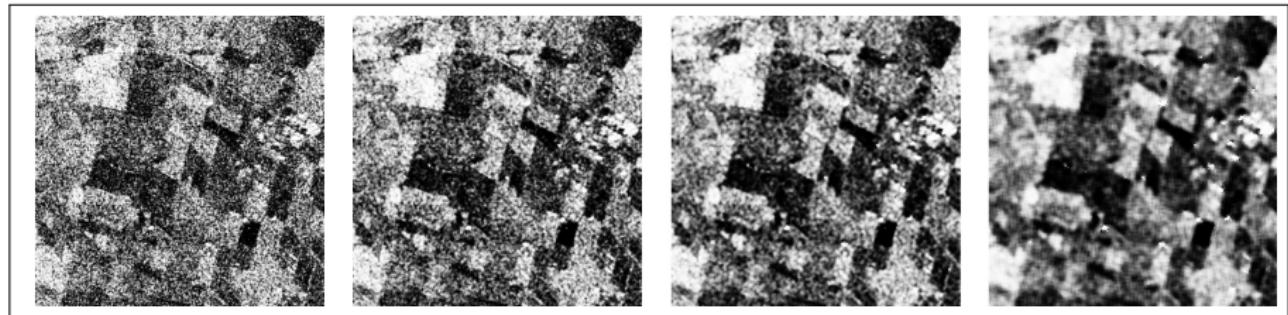
# Resultados y Ejemplos

- Kuan
- Isodata
- Frost + K-Means
- Video de ejemplo de Isodata.
- PyRadar Frost vs Envi Frost

# Resultados y Ejemplos

- Kuan
- Isodata
- Frost + K-Means
- Video de ejemplo de Isodata.
- PyRadar Frost vs Envi Frost
- PyRadar Isodata vs Envi Isodata

## Kuan



Filtro de Kuan con *winsize* 3, 5, 7, 11 y coeficiente de variación del ruido  $C_u$  de 0.3.

# Isodata

- Iteraciones máximas: 1000,
- $P = 4$
- $\theta_M = 10, \theta_S = 1, \theta_C = 20, \theta_O = 0,05.$

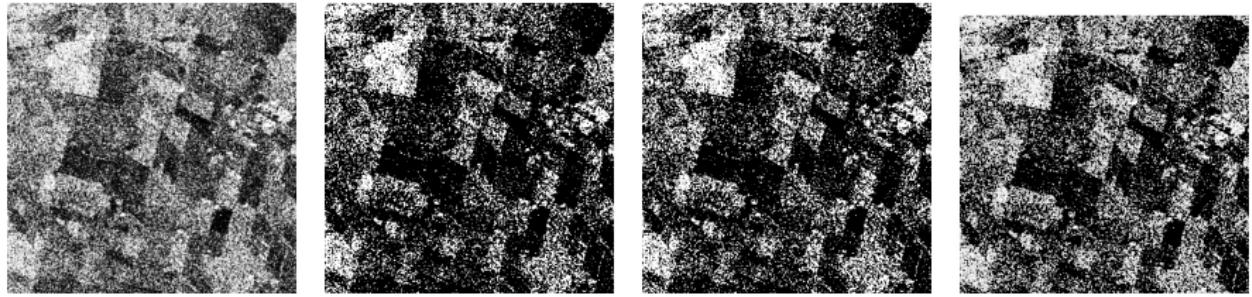
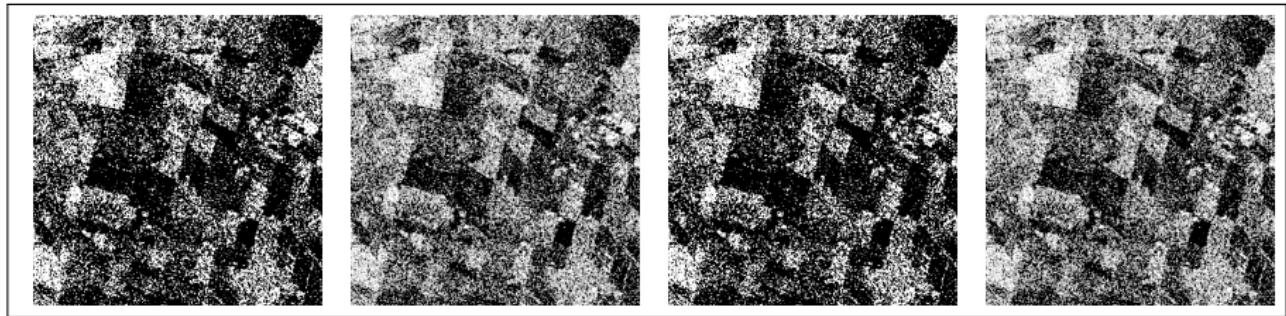


Imagen original, Isodata con 3, 5, 7 clases.

# Frost + K-Means

Parámetros:

- Primera imagen: damp 0.10, clases 3,
- Segunda imagen: damp 0.10, clases 5,
- Tercera imagen: damp 1.0, clases 3,
- Cuarta imagen: damp 1.0, clases 5.

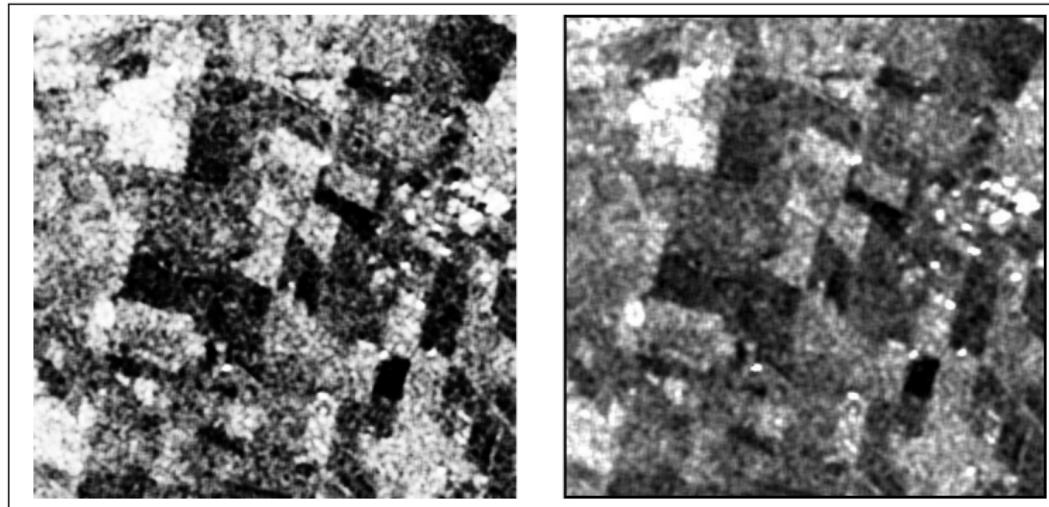


# Video de ejemplo de Isodata.

# Frost

Parámetros:

- window size: 7
- damping factor: 1.0

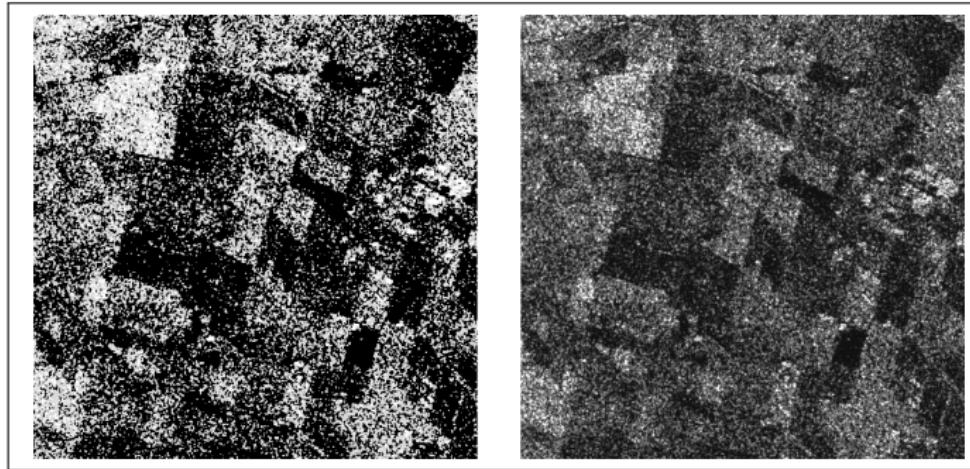


PyRadar a la izquierda, ENVI a la derecha.

# Isodata

Parámetros:

- Clases 8,
- Iteraciones máximas: 1000,
- $P = 4$
- $\theta_M = 10, \theta_S = 1, \theta_C = 20, \theta_O = 0,05$ .



PyRadar vs Envi

¡Vamo' a lo bife!

¿Preguntas?

# ¡Muchas Gracias!

— Matías y Joaquín.



— Fin.