[dstl] The Science Inside

Implementation of Adaptive Kernel Kalman Filter in Stone Soup

James S. Wright, James R. Hopgood, Mike E. Davies, Ian K. Proudler, Mengwei Sun

Sensor Signal Processing for Defence (SSPD) Conference, Edinburgh, 2023

Ministry of Defence

Introduction

- Adaptive Kernel Kalman Filter
- This is the first open source implementation of the AKKF

Introduction: Outline

- Stone Soup: Tracking and State Estimation Framework
- Adaptive Kernel Kalman Filter
- Components
- Examples
- Conclusions

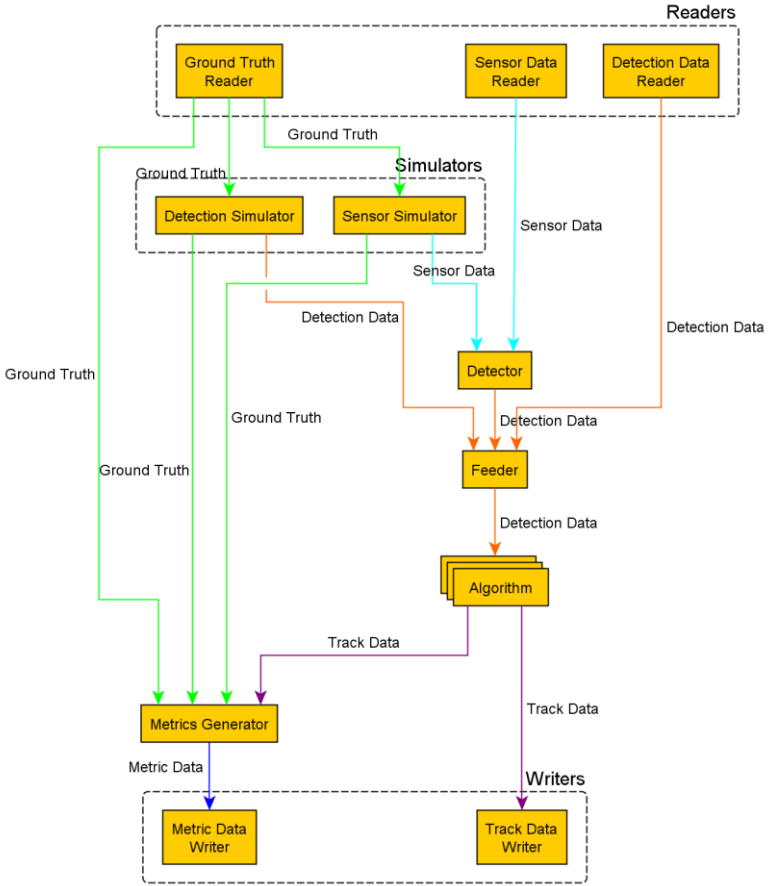
What is Stone Soup?

Algorithms | Data | Sensor models | Simulators | Metrics

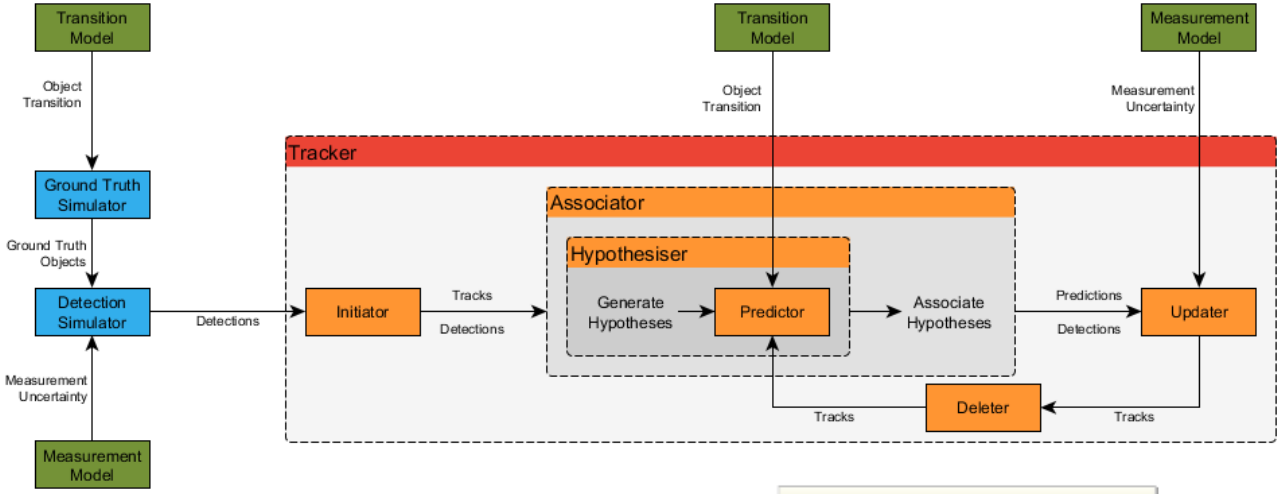
Framework (Open Source)

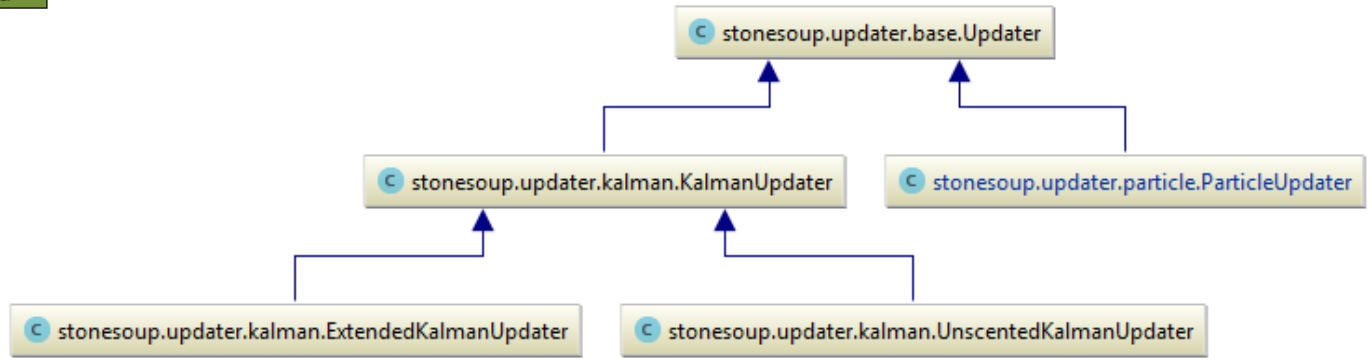
Stone Soup: Components

- Readers
- Simulators
- Feeders
- Algorithmic Components
  - Predictors
  - Updaters
- Metrics
- Writers

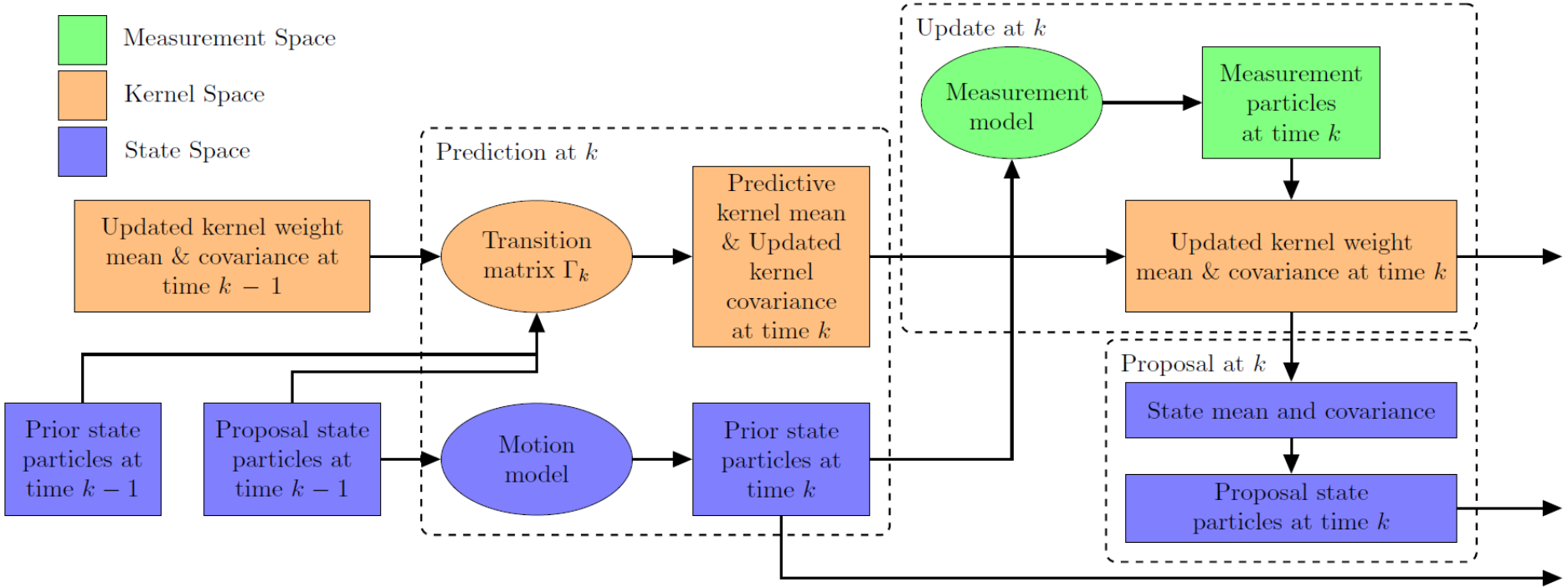


Stone Soup: Algorithmic Components





Adaptive Kernel Kalman Filter (AKKF)



Components

- States
  - `KernelParticleState`
- Kernels
  - `Kernel, QuadraticKernel, QuarticKernel, GaussianKernel`
- Predictors
  - `AdaptiveKernelKalmanPredictor`
- Updaters
  - `AdaptiveKernelKalmanUpdater`

`KernelParticleState`

Inherits the functionality of `ParticleState` and adds the kernel covariance defined by the following equations

$$S_k^- = \Gamma_k S_{k-1}^+ \Gamma_k^T + V_k$$

$$S_k^+ = S_k^- - Q_k G_{yy} S_k^-$$

Components: Kernels

- `Kernel`

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}}\, \boldsymbol{x}'$$

- `QuadraticKernel`

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\alpha\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \ c)^2$$

- `QuarticKernel`

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\alpha\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \ c)^4$$

- `GaussianKernel`

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}\right)$$

Components: Predictors and Updaters

[dstl] The Science Inside

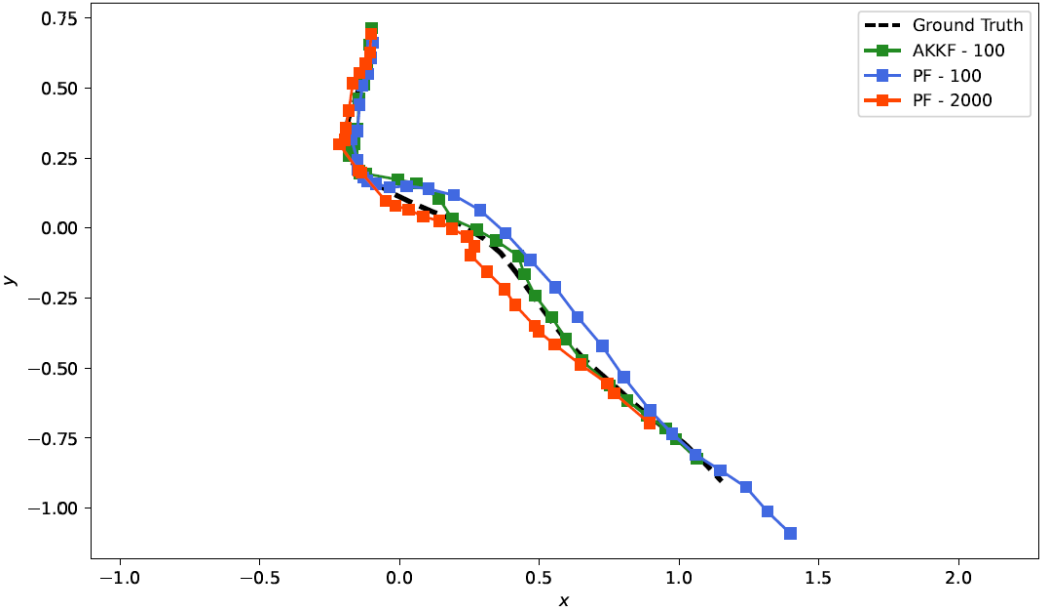
`AdaptiveKernelKalmanPredictor`

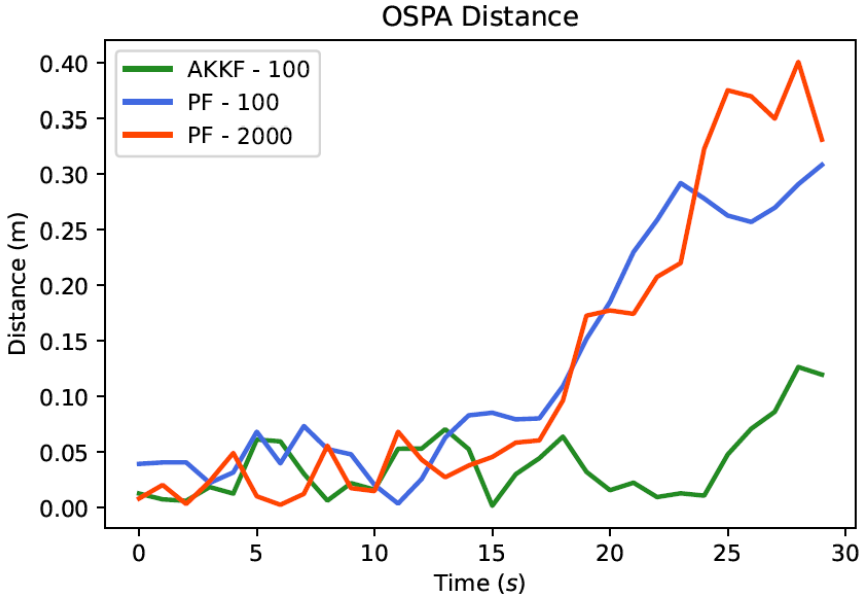
Inherits the functionality of `KalmanPredictor` and adds the functionality of translating between state space and kernel space.

`AdaptiveKernelKalmanUpdater`

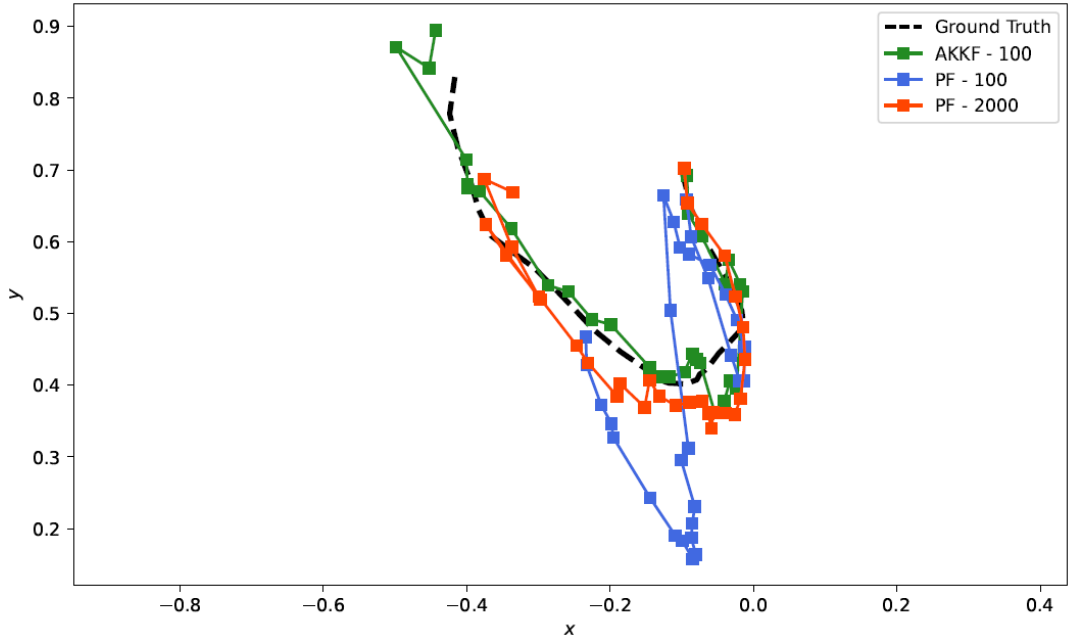
Inherits the functionality of `KalmanUpdater` and adds the functionality of translating between measurement space and kernel space.

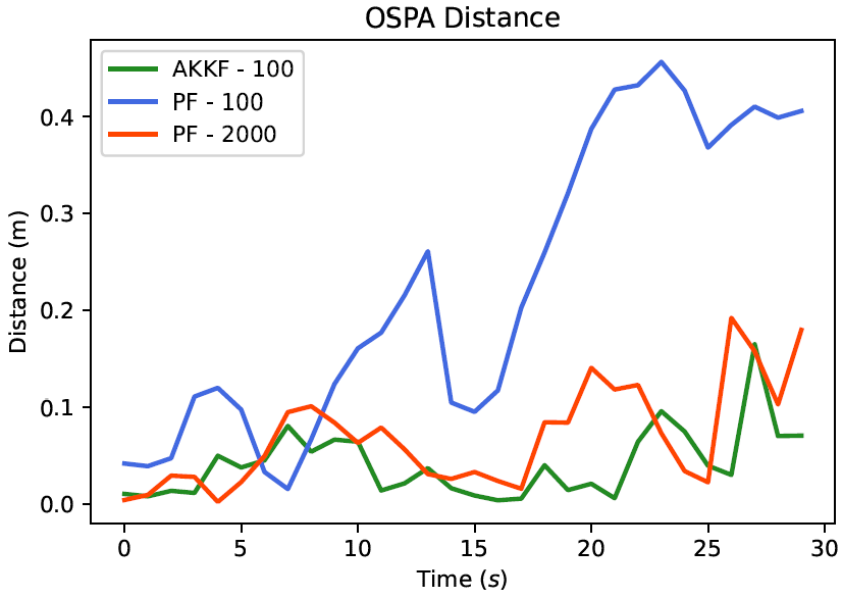
Example 1


Ground Truth
AKKF - 100
PF - 100
PF - 2000
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

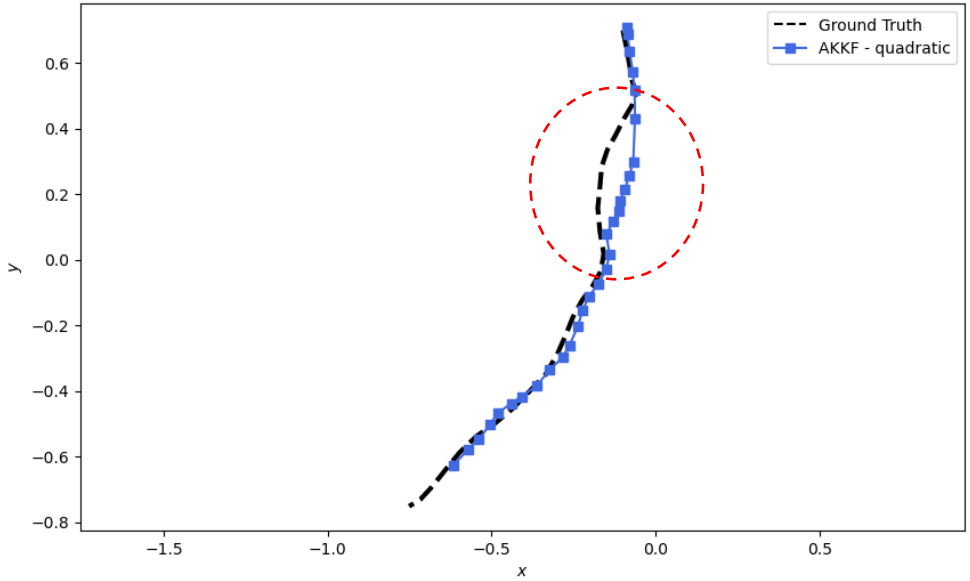

OSPA Distance
AKKF - 100
PF - 100
PF - 2000
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Distance (m)
0
5
10
15
20
25
30
Time (s)

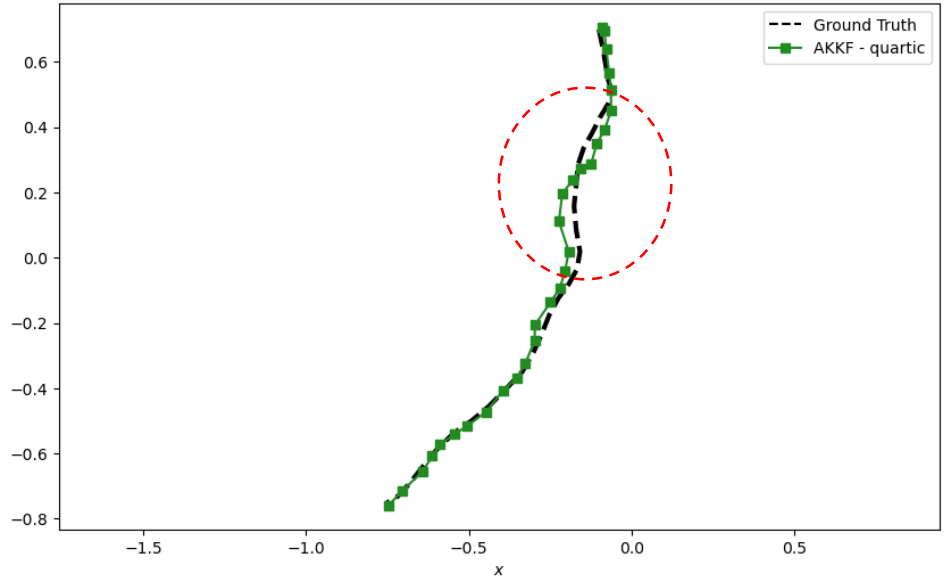
Example 2


Ground Truth
AKKF - 100
PF - 100
PF - 2000
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
y
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
x


OSPA Distance
AKKF - 100
PF - 100
PF - 2000
Distance (m)
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
30
Time (s)

Example 3: Which Kernel?





Conclusions

- AKKF outperforms PF for a given number of particles
- We provided an open source implementation of the AKKF in Stone Soup
- Via inhertance, modularity, and encapsulation, we have demonstrated that Stone Soup is an efficient way of enacting, testing and demonstrating the efficacy of novel tracking algorithms
- We have shown a method of integrating newer algorithms into Stone Soup and a workflow to support other algorithms being implemented in Stone Soup

Get in touch:

https://github.com/dstl/Stone-Soup **StoneSoup@dstl.gov.uk**

[dstl] The Science Inside

Discover more