Reviving Timing-based localization study

M. Ohno

May 26th 2020

Part 1. CCF analysis algorithm updating project
Part 2. Application of the machine learning approach

Part 1. Update CCF analysis algorithm

M. Ohno and J. Rípa

May 26th 2020

Current localization problem

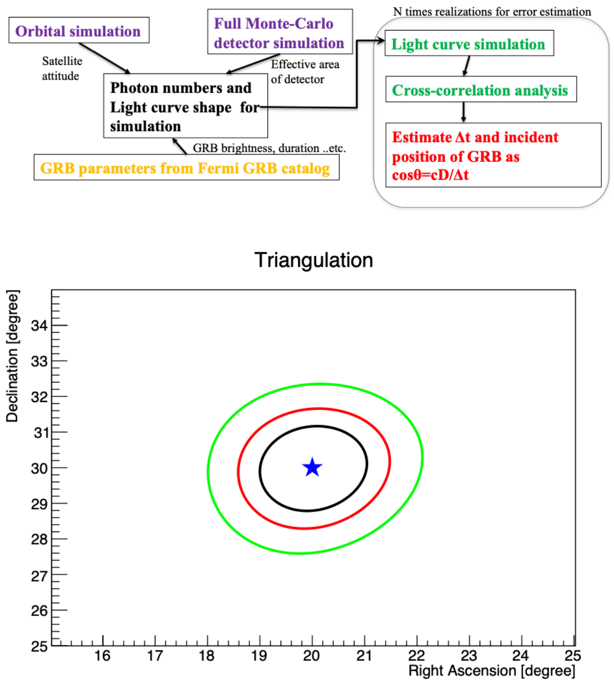
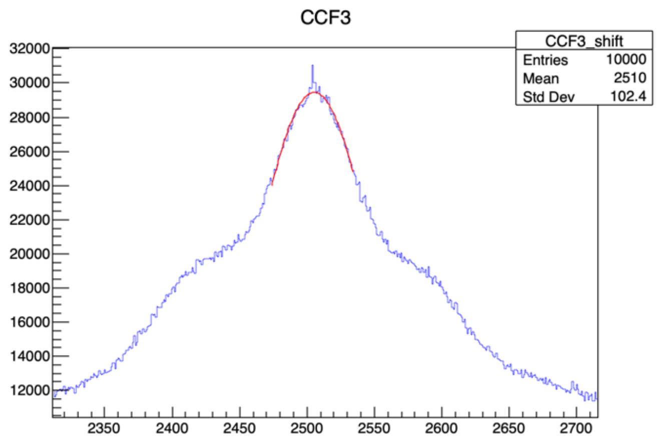
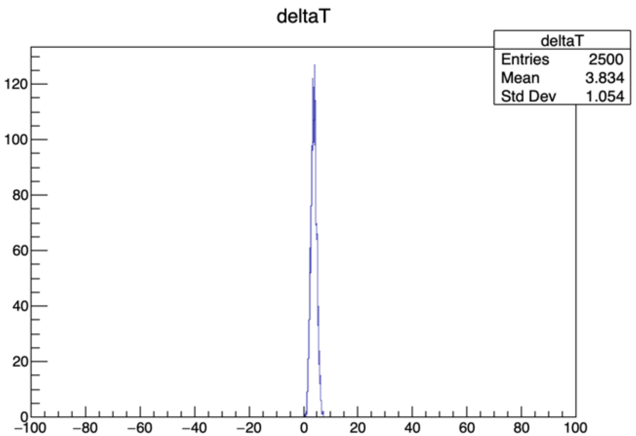


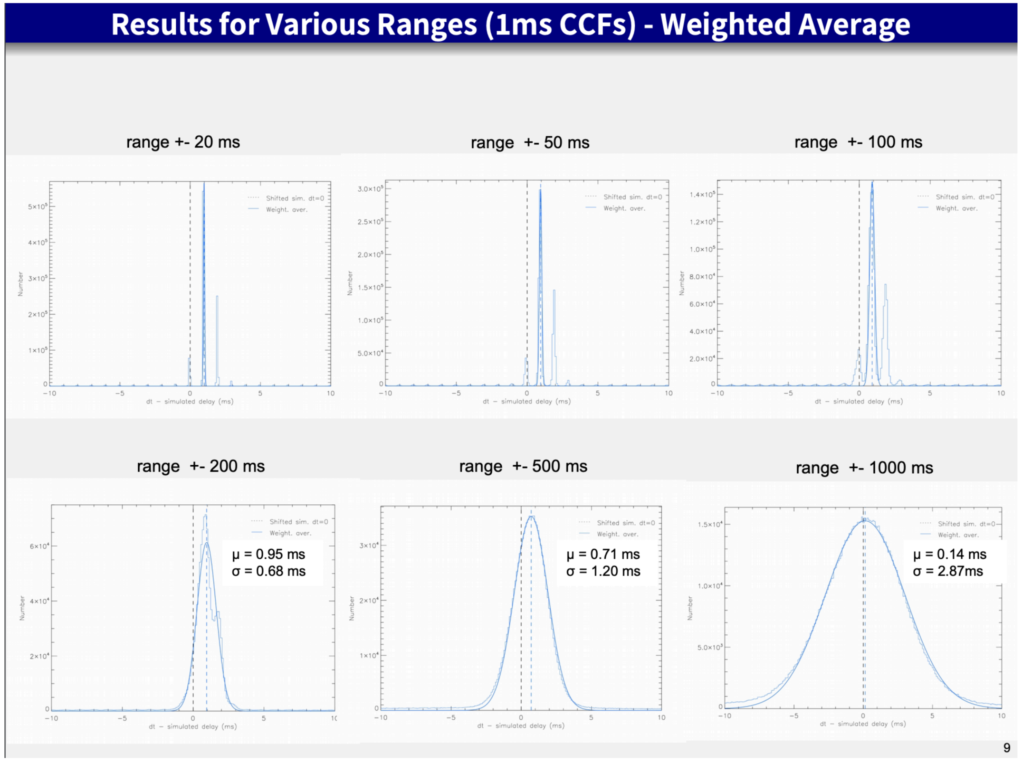
Fig. 2. top: a flow-chat of the localization simulation framework in this study. bottom: the example of the localization by using 9 satellites combination for the bright short GRB 090227772. A star marker shows the input position (R.A.=20, Dec=30 degree), and black, red and green contour is obtained confidence regions correspond to 1, 2 and 3 sigma significance.

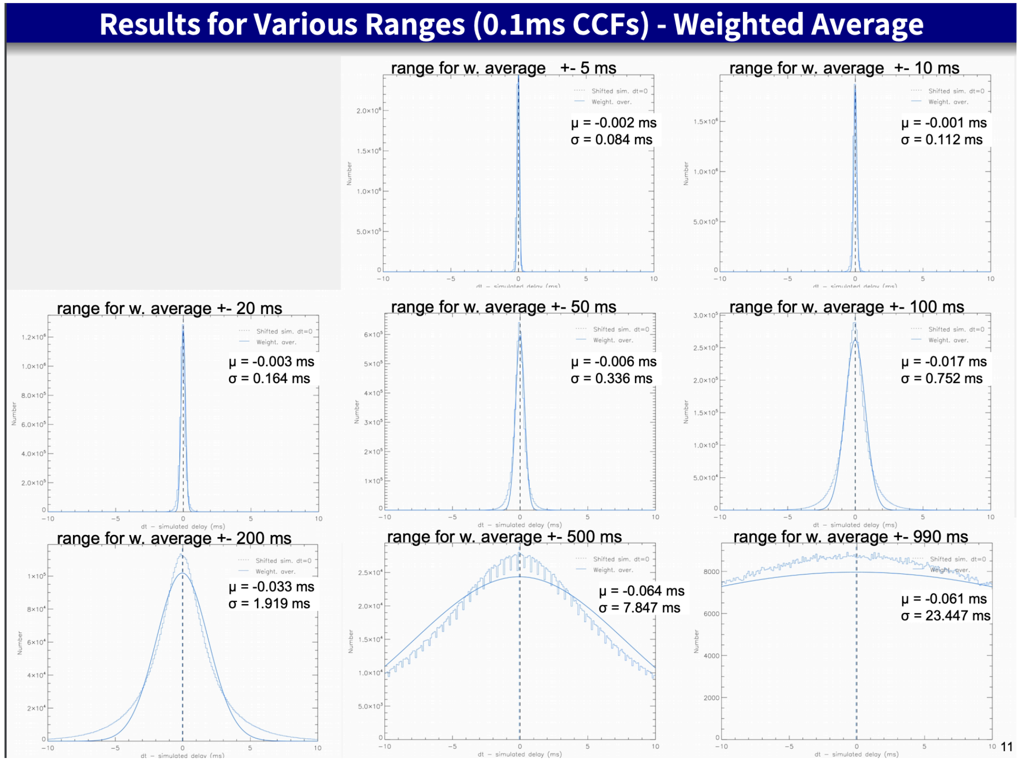




- ✓ Current localization uncertainty is mainly caused by the uncertainty of the estimated time-difference (delta-t)
- ✓ The CCF peak is obtained by the gaussian function fitting around the CCF maximum
- ✓ Delta-t uncertainty is limited to be ~1ms even for the brightest short GRB ➔ current best localization error is still ~ 1deg
- ✓ Human-eyes verification is mandatory for such fitting procedures
- ✓ Other better algorithm to determine the CCF peak ?

Gaussian fitting vs weighted mean

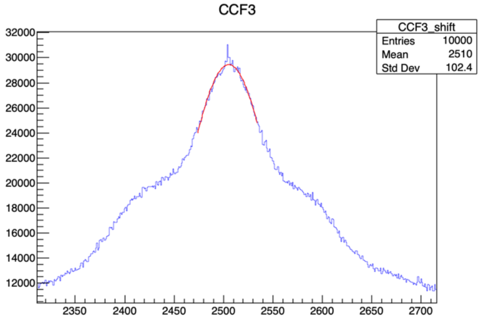




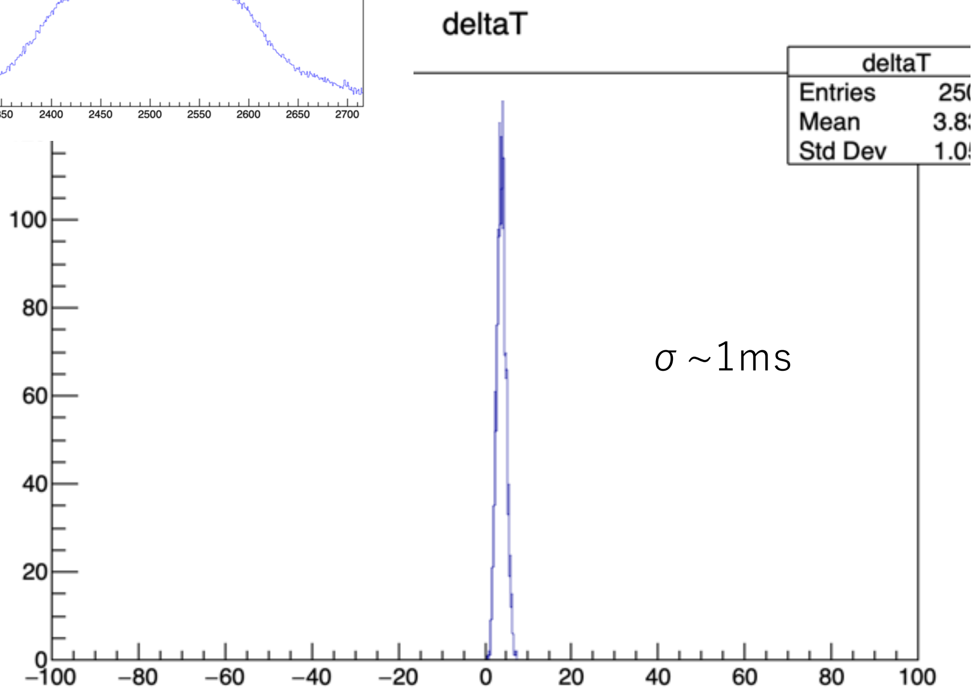
From Jakub-san's analysis, weighted mean could have < 1ms delta-t uncertainty (1-sigma), let's check it !

Analysis samples

- Only one bright short GRB090227772 is tested
- 2500 simulations with a fixed given delta-t (4ms)
- 1 ms time resolution is checked so far
- Peak is estimated by (excluding artificial one-bin spike)
  - gaussian fitting around the maximum bin (+/-30ms)
  - weighted mean around the maximum bin (+/-30ms)
- The maximum bin is estimated by
  - a simple maximum bin by the original CCF
  - a maximum bin excluding artificial one-bin spike



Gaussian fitting

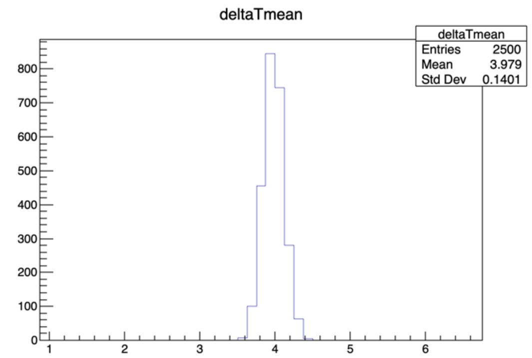


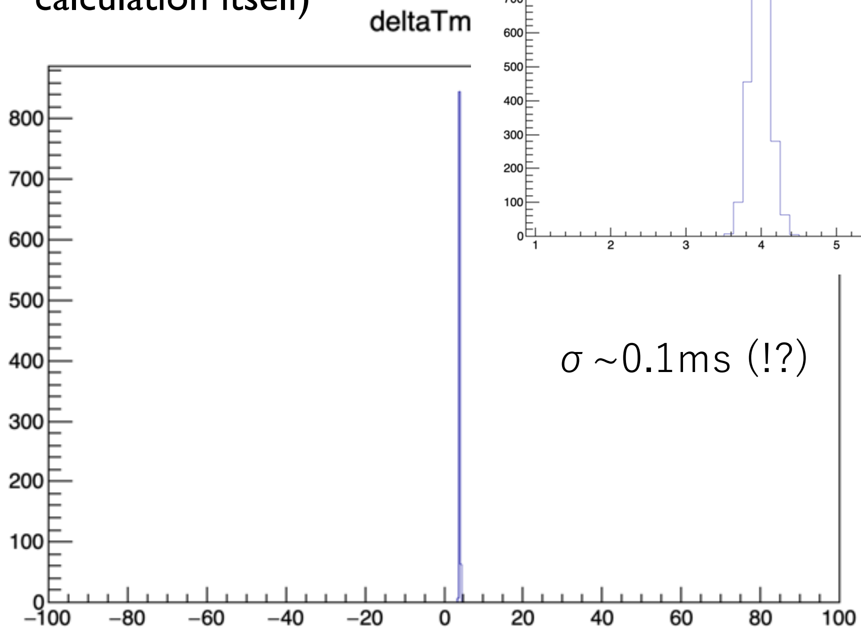
σ ~1ms

Delta-T

Weighted Av. around
"simple" maximum bin
== "Fixed calculation center"
(strong spike is excluded
from weighted average
calculation itself)

Zoom



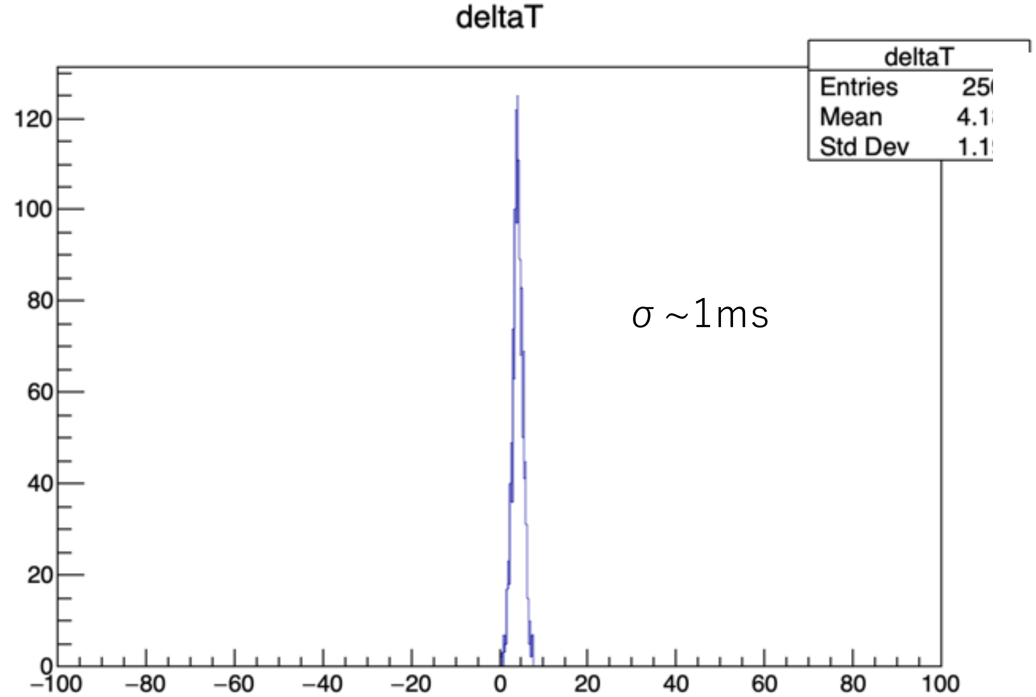


σ ~0.1ms (!?)

Delta-T

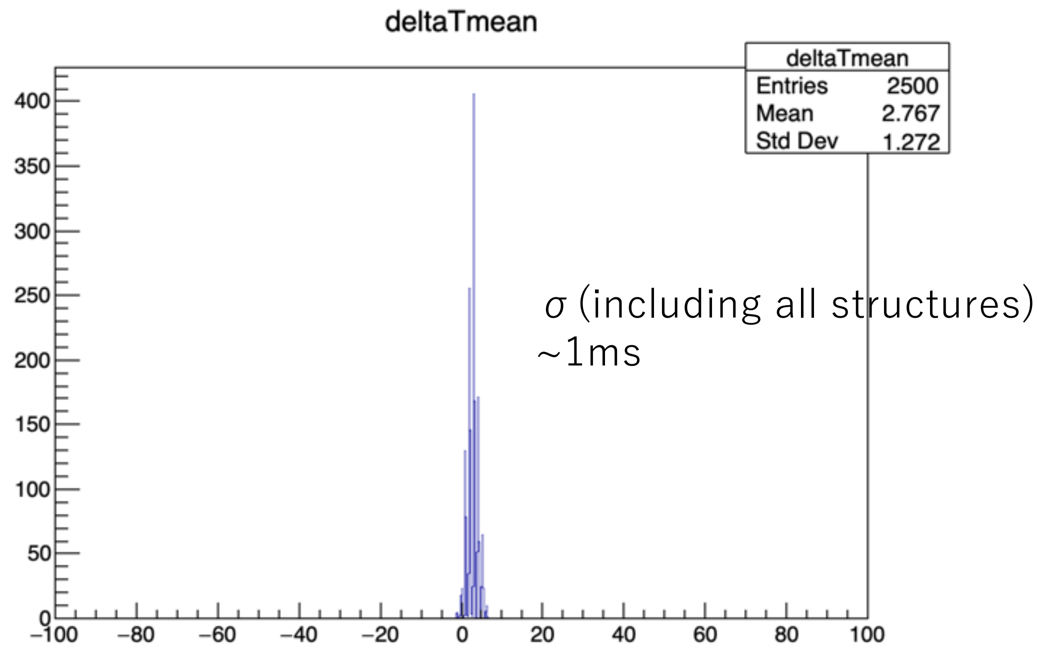
Analysis range : +/-30ms (typical minimum variability timescale Bhat+13)

Gaussian fitting



Delta-T

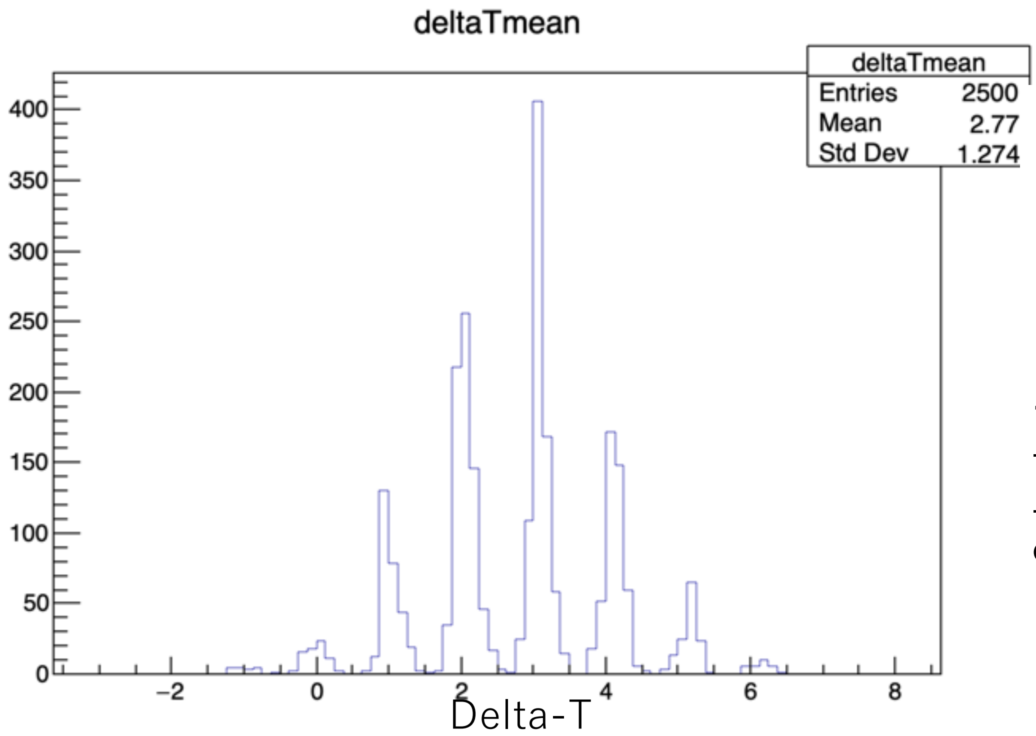
Weighted Av. The maximum bin is estimated exclude the strong spike bin



Delta-T

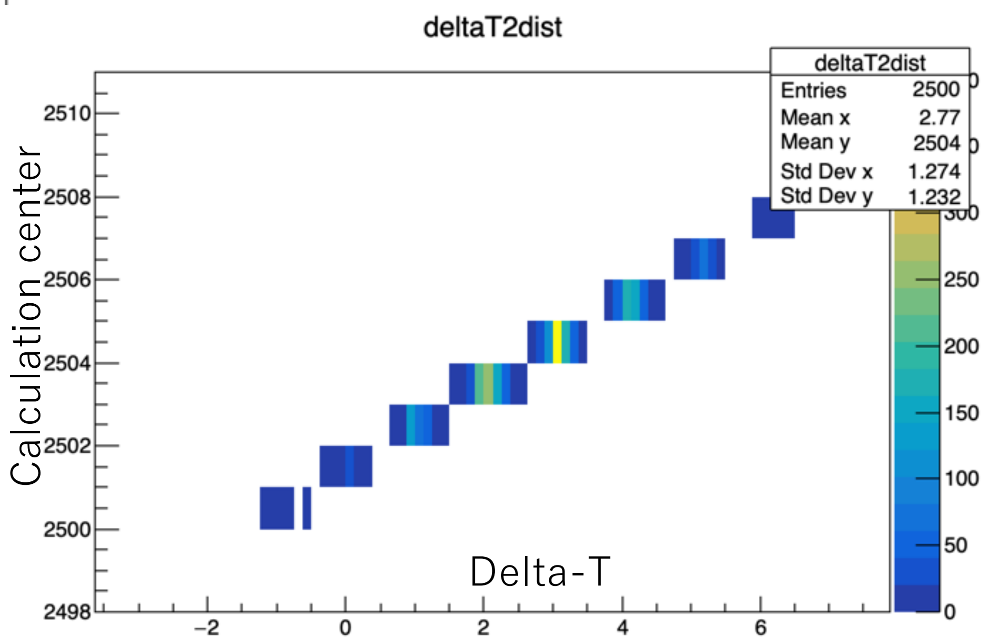
Analysis range : +/-30ms (typical minimum variability timescale Bhat+13)

Zoomed up: Weighted Av. The maximum bin is estimated exclude the strong spike bin



Each sub-structure shows similar shape to the "fixed calculation center" case.
Including sub-structure, entire width of distribution is similar to the gaussian fitting case

To be continued this work
- use 0.1 ms LC
- iterative process to determine the analysis center



Each sub-structure belong to the different calculation center

Timing based localization by the machine learning approach

M. Ohno, Y. Ichinohe, R. Jakub, and N. Werner

May 26th, 2020 v0: simple time delay estimation

Motivation

- CCF based time-delay analysis is a standard way for the timing-based localization
- Problem for the CCF analysis are

1) variation of the CCF shape makes the framework automated difficult (need human-eyes analysis and validation)

2) current gaussian fitting approach limits the accuracy ~1ms (corresponds to ~1 deg localization error)

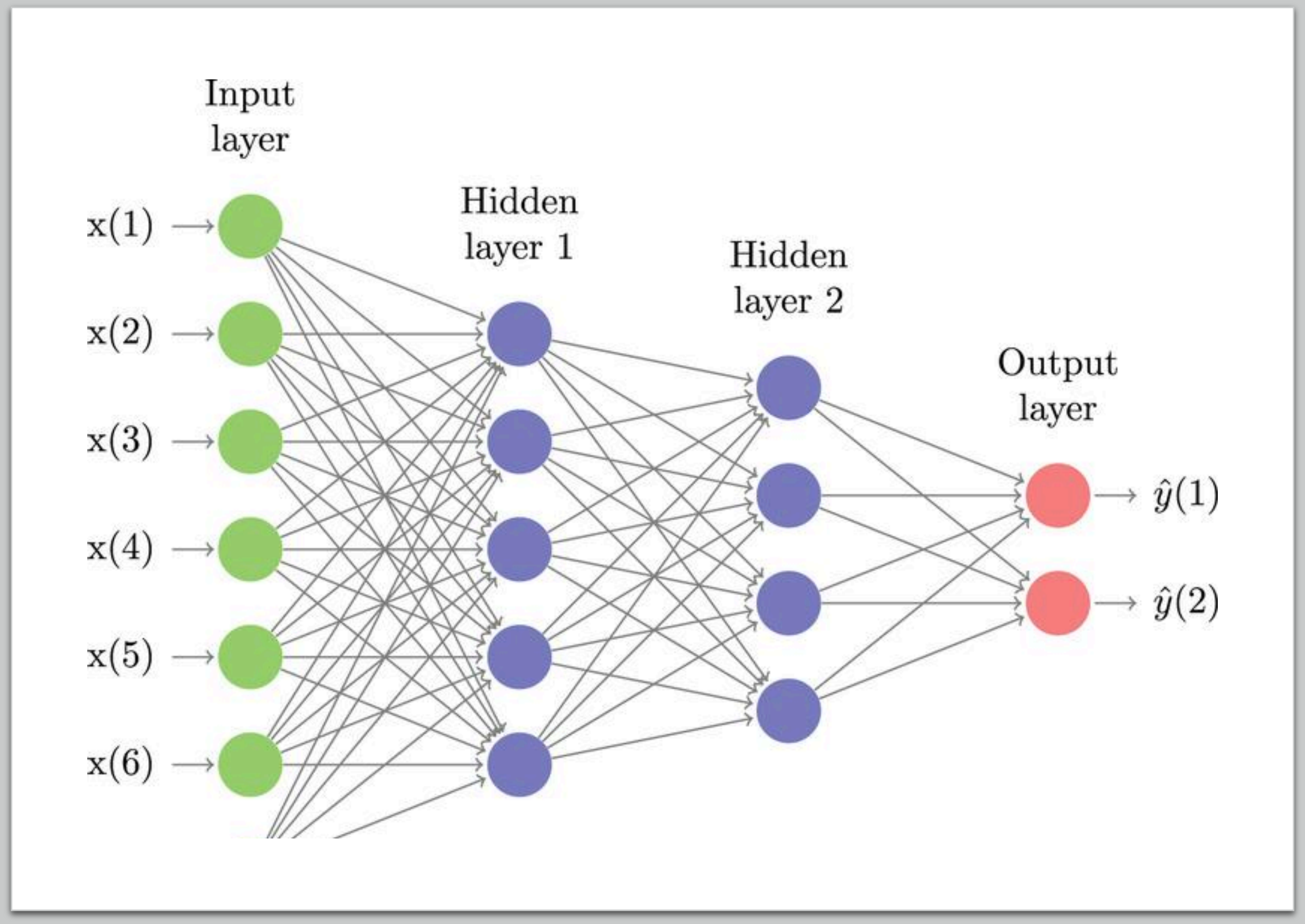
- Machine learning approach ?

  - helpful to automation ?
  - any possibilities to improve the time-delay estimation ?

Goal of the concept

input layer = light curve from multiple satellites

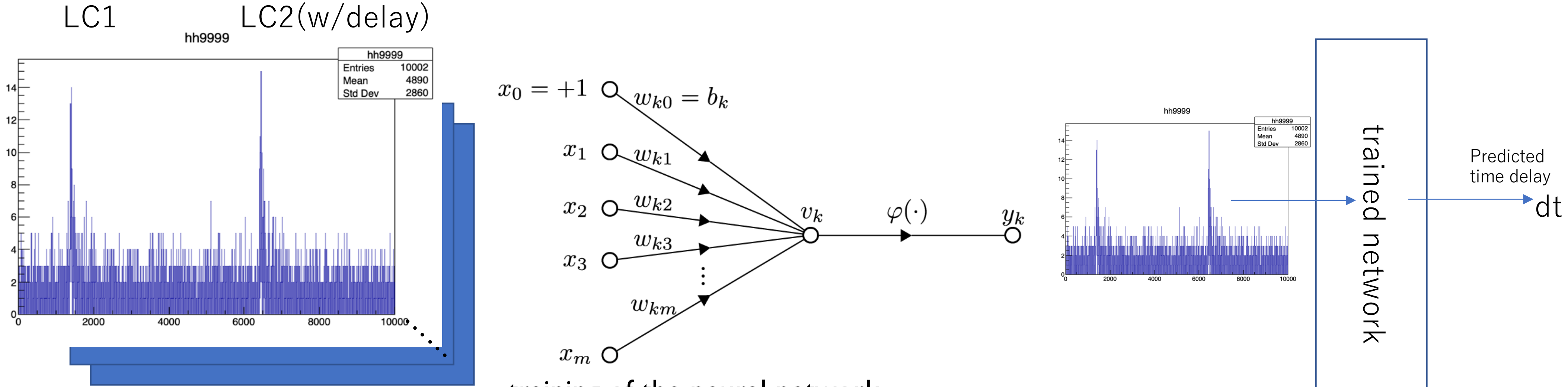
output layer = parameter vector that we want to know (RA, Dec, flux .. Etc.)



Is this concept realistic ? Let's start from a simple estimation !

A simple time-delay estimation

- Input layer : two light curves with a give time delay (1-D vector)
  simulated from the same light curve (GRB090227772)
- Output layer : predicted time delay
- training : large number sets of two light curves with a random time delay (-50ms to +50ms)



training of the neural network
(weight/activation function)

Tensorflow as a Keras backend

- Tensorfow : open source library for the deep learning
- Keras : python based open source neural network library for quick experiment of the deep neural network learning which supports Tensorflow as the backend
- I need more "deep learning" to explain what those framework are, and how they are working !

```
model=Sequential()
model.add(Dense(units=48, input_shape=(orig_dim,),activation='sigmoid'))
model.add(Dense(units=24,activation='sigmoid'))
model.add(Dense(units=1))
```

Define sequential hidden layers with activation function

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

Define how to optimize the parameters (loss function, optimizer sgd: stochastic gradient decent)

```
history=model.fit(xtrain,ytrain,epochs=nepochs,batch_size=30,validation_data=(xvalid,yvalid))
```

Run training using xtrain (light curve) and ytrain (answer:dt) with some iteration parameters (ephocs)

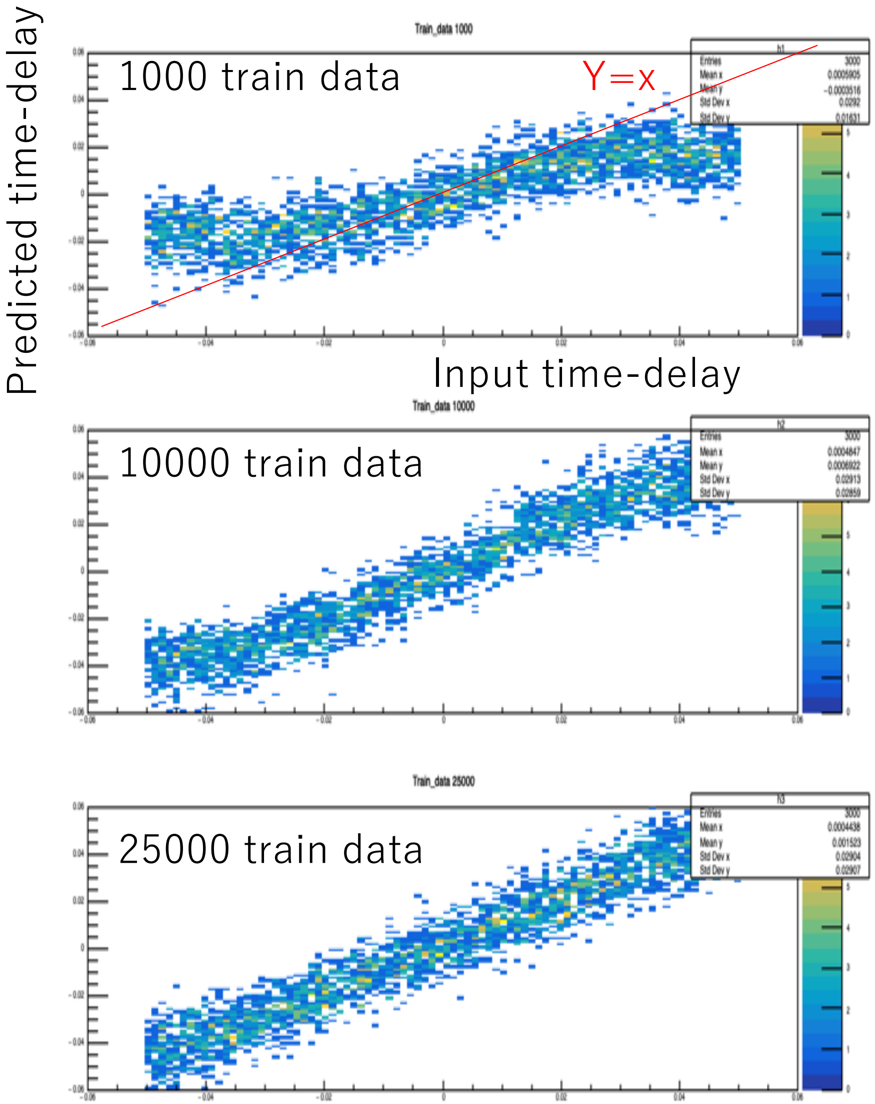
Training the network !

```
ターミナル — -tcsh — Basic — ⌥⌘2
8d20 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2020-05-26 02:20:26.265088: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor devic
e (0): Host, Default Version
Train on 5000 samples, validate on 3000 samples
Epoch 1/50
5000/5000 [==============================] - 1s 125us/step - loss: 0.0067 - val_loss: 8.4776e-04
Epoch 2/50
5000/5000 [==============================] - 0s 97us/step - loss: 8.6877e-04 - val_loss: 7.9108e-04
Epoch 3/50
5000/5000 [==============================] - 0s 98us/step - loss: 8.0999e-04 - val_loss: 7.4528e-04
Epoch 4/50
5000/5000 [==============================] - 0s 97us/step - loss: 7.5696e-04 - val_loss: 6.8836e-04
Epoch 5/50
5000/5000 [==============================] - 0s 96us/step - loss: 7.0569e-04 - val_loss: 6.4489e-04
Epoch 6/50
5000/5000 [==============================] - 0s 97us/step - loss: 6.6064e-04 - val_loss: 6.1185e-04
Epoch 7/50
5000/5000 [==============================] - 0s 95us/step - loss: 6.1716e-04 - val_loss: 5.6381e-04
Epoch 8/50
5000/5000 [==============================] - 0s 99us/step - loss: 5.7875e-04 - val_loss: 5.3748e-04
Epoch 9/50
5000/5000 [==============================] - 0s 97us/step - loss: 5.4333e-04 - val_loss: 4.9581e-04
Epoch 10/50
5000/5000 [==============================] - 0s 96us/step - loss: 5.0910e-04 - val_loss: 4.6584e-04
Epoch 11/50
5000/5000 [==============================] - 0s 96us/step - loss: 4.7810e-04 - val_loss: 4.3833e-04
Epoch 12/50
5000/5000 [==============================] - 0s 96us/step - loss: 4.4944e-04 - val_loss: 4.1285e-04
Epoch 13/50
5000/5000 [==============================] - 0s 97us/step - loss: 4.2409e-04 - val_loss: 3.8886e-04
Epoch 14/50
5000/5000 [==============================] - 0s 96us/step - loss: 4.0019e-04 - val_loss: 3.6735e-04
Epoch 15/50
5000/5000 [==============================] - 0s 97us/step - loss: 3.7705e-04 - val_loss: 3.4700e-04
Epoch 16/50
```

Training data : 5000 set
Iterative number : 50 epochs

The loss function estimate value decreasing
➔ Closer to the input value

Here, various parameters are investigated
- Amount of the training data
- Amount of the iterative sequence
- time-resolution of the input data



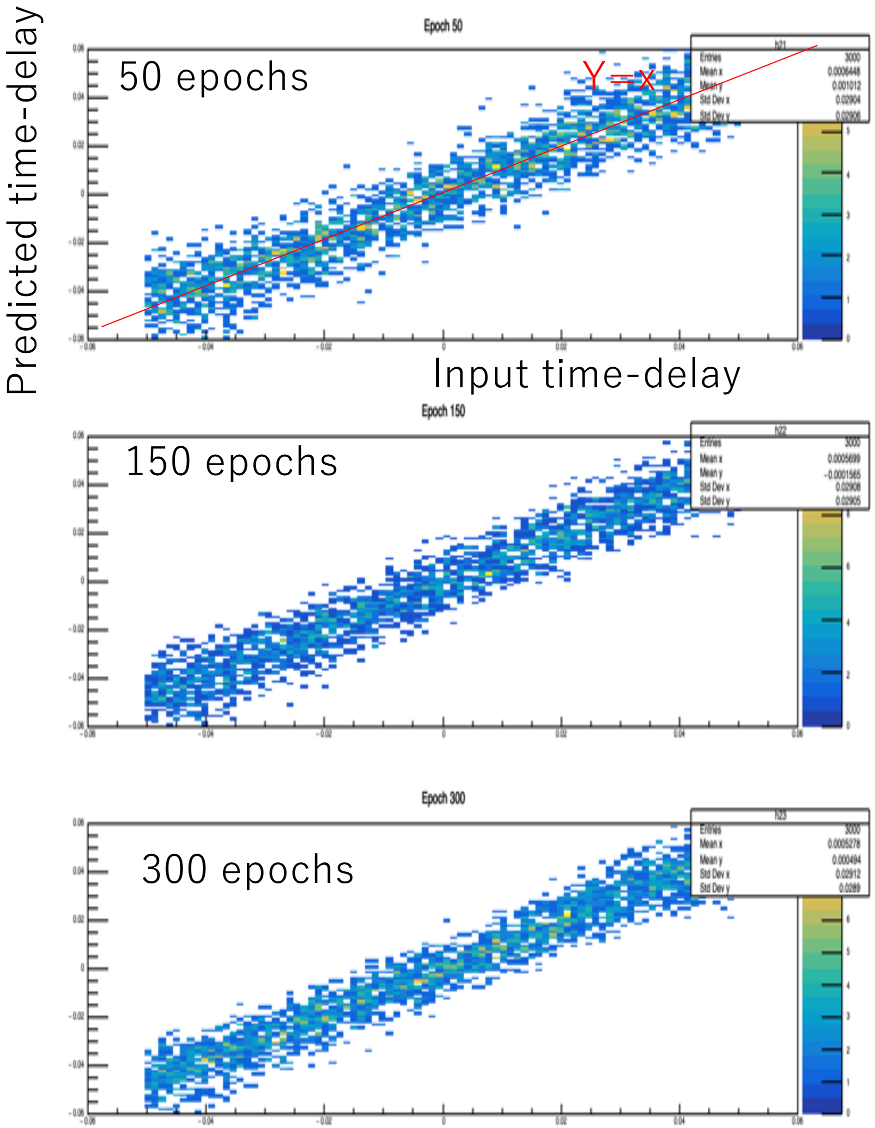
Results 1: train data set dependency

- Input light curve : 1 ms resolution
- Iterative number is fixed to be 100

- Systematic deviation is seen for the smaller train data result

- 25000 train data gives almost straight result

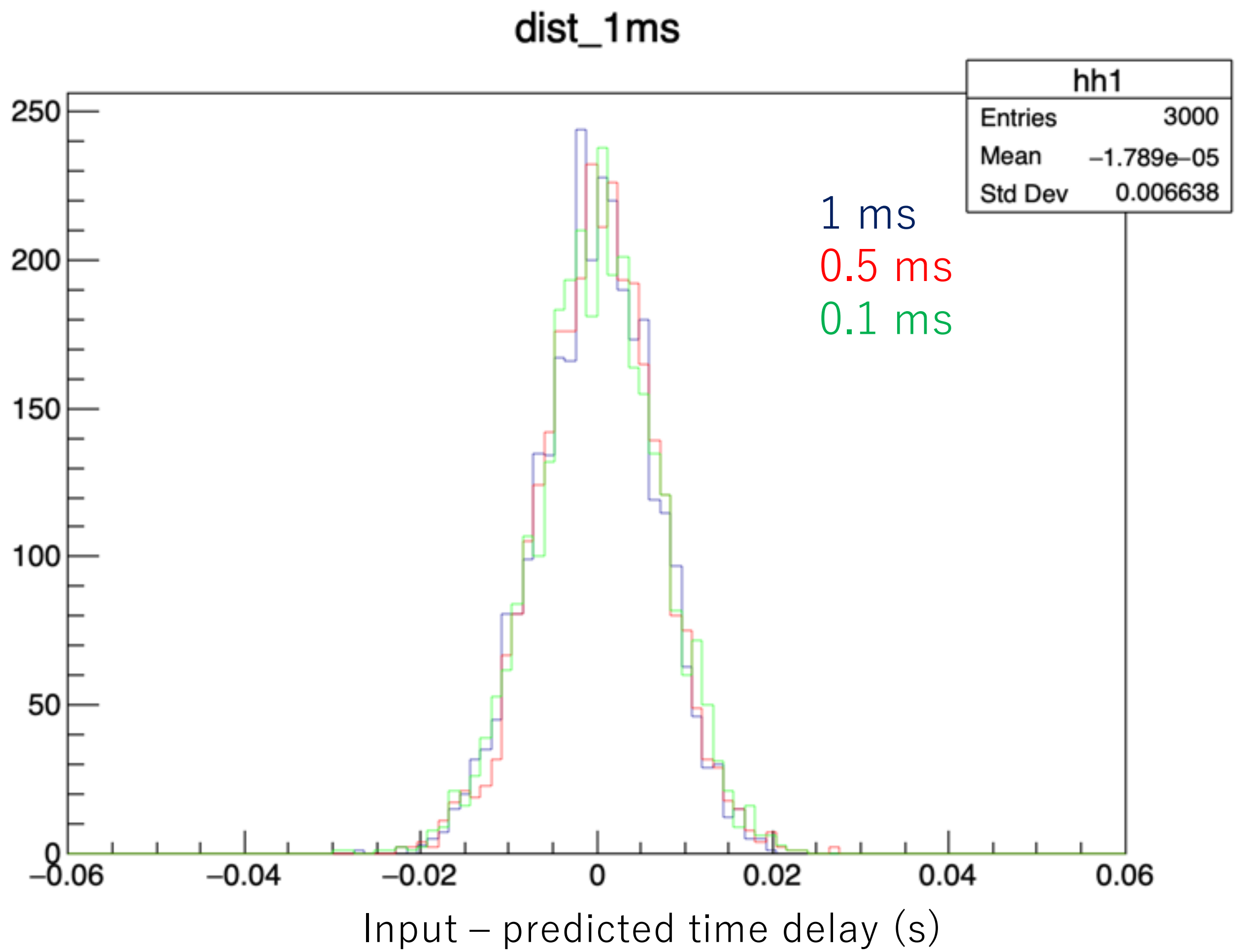
Result2: iterative number dependency



- Input light curve : 1 ms resolution
- Train data number is fixed to be 25000

- Larger iteration gives a narrower distribution (higher accuracy !)

Result 3 : time-resolution and current deltaT estimation uncertainty



- Iterative number is fixed to be 300
- Train data number is fixed to be 25000

- Accuracy of the predicted time delay does NOT depend on the time-resolution of the input light curve, interesting…

- Current delta-t estimation uncertainty is ~6 ms, which is still much worse than the CCF approach (~1ms)

- Further investigation of the training approach could improve the result ? Under discussion with Yuto !