Published by IOP Publishing for Sissa Medialab

Received: *June 3, 2021*
Revised: *July 16, 2021*
Accepted: *July 21, 2021*
Published: *August 20, 2021*

TECHNICAL REPORT

2021 JINST 16 T08016

Efficient and versatile toolbox for analysis of time-tagged measurements

Z. Lin,[a,b,1] **L. Schweickert,**[a,1,*] **S. Gyger,**[a] **K.D. Jöns**[a,2] **and V. Zwiller**[a]

[a] *Department of Applied Physics, Royal Institute of Technology, Roslagstullsbacken 21, 114 21 Stockholm, Sweden*
[b] *School of Precision Instrument and Opto-electronics Engineering, Tianjin University, Weijin Road 92, Tianjin, China*

E-mail: lucassc@kth.se

Abstract: Acquisition and analysis of time-tagged events is a ubiquitous tool in scientific and industrial applications. With increasing time resolution, number of input channels, and acquired events, the amount of data can be overwhelming for standard processing techniques. We developed the **E**xtensible **T**ime-tag **A**nalyzer (ETA), a powerful and versatile, yet easy to use software to efficiently analyze and display time-tagged data. Our tool allows for flexible extraction of correlation from time-tagged data beyond start-stop measurements that were traditionally used. A combination of state diagrams and simple code snippets allows for analysis of arbitrary complexity while keeping computational efficiency high.

Keywords: Data processing methods; Detector control systems (detector and experiment monitoring and slow-control systems, architecture, hardware, algorithms, databases); Simulation methods and programs; Analysis and statistical methods

ArXiv ePrint: 2105.15117

[1] Authors contributed equally.
[2] Current affiliation: Department of Physics, Paderborn University, 33098 Paderborn, Germany.
[*] Corresponding author.

https://doi.org/10.1088/1748-0221/16/08/T08016

Contents

2021 JINST 16 T08016

1 Introduction

Extracting correlation from time resolved data [1] gives insights into the dynamics of a system under study over more than 18 orders of magnitude, from picoseconds to hours, in a single experiment. This makes it one of the most powerful tools for data analysis widely used in the sciences. Time resolution better than 8 picoseconds is already available at the time of writing [2] and can be expected to reach the femtosecond range in the near future [3]. In optics, the correlation among photon detection events is often analyzed to investigate underlying physical processes [4, 5]. Examples include (i) light detection and ranging (LIDAR), where time-of-flight measurements, a subclass of correlation measurements, provide the distance to a reflective or scattering medium [6], (ii) random number generation where the timing and probability of the events generate random values [7], and (iii) characterization of quantum emitter properties and determination of the number of emitters under study [8]. Correlation measurements are also required to characterize entangled states, be it well-known two-photon entangled states [9, 10], more complex multi-photon entangled states such as GHZ [11] or cluster states [12].

As an example, the usual experimental setup in quantum optics is based on the well-known Hanbury Brown and Twiss (HBT) experiment [13], schematically shown in figure 1: a stream of photons is directed at a beam splitter with click detectors at each output. Here, the use of a beam splitter allows for detection events to be obtained at shorter time intervals than the detectors' dead time. Using a single detector limits the time resolution of the system to the detector dead time but can still reveal a correlation on slower time scales [14].

Correlation between two click-detectors was historically measured with a time-to-amplitude converter, where one detector starts a timer and the other detector stops it again, generating a time interval value, as illustrated in figure 1a. After accumulating a significant number of them, these time intervals can be plotted in a histogram. The recent advent of time-tagging techniques [15]

for photon detection events with timing resolution comparable to the coherence and lifetimes of quantum emitters offers an alternative to the well established start-stop histograms obtained directly with analogue timing electronics. In time-tagging, fast electronics record the occurrence of each detection event with respect to an absolute time t_0, generating a timetable of events from all detectors, as illustrated in figure 1b. Rudimentary software offerings from manufacturers of these time-tagging devices, however, often require that the analysis method is selected from a predefined list of options ahead of time and only the resulting histogram is stored. This approach does not allow for more complex experiments, like e.g. entanglement swapping [16, 17], quantum key distribution [18], continuous variable entanglement [19], spin-spin entanglement [20] or teleportation [21], and only a single method of analysis can be chosen per experiment.

When making full use of modern time-tagging hardware, instead of committing to an analysis method before the start of the experiment, all timing information can be saved to disk. The resulting time-tagged files can be analyzed after the measurement in various ways to extract correlation between the recorded channels. Depending on the experiment, time-tagged files can require terabytes of storage space. This makes data analysis a major hurdle and specialized software is needed to extract useful information in a reasonable time. Therefore, efficient correlation extraction and processing of large data sets is still a widely encountered challenge in a broad range of applications and research fields.

The importance of user-defined analysis using time-tagged data becomes apparent when looking at an example in more detail: when logging the arrival times of single photons emitted by a quantum dot along with the laser excitation events, we can extract the exciton lifetime, the biexciton lifetime, the exciton emission auto-correlation, the biexciton auto-correlation, the time evolution of count rates for exciton and biexciton, two-time correlation as well as cross-correlation between exciton and biexciton. Using time-tagging, these results can be extracted by analyzing data from only one experiment, not only saving time but also offering more reliable results since the data were acquired under identical conditions [22].

We created a versatile toolbox, **E**xtensible **T**ime-tag **A**nalyzer (ETA),[1] for analysis of time-tagged data, enabling extraction of a wide range of information from one experiment with high efficiency.

So far, a modeling language designed to intuitively allow the specification of a particular time-tag analysis method to be executed by software has been elusive. Researchers tend to use a familiar general-purpose programming language to solve the problem at hand. The result is a clustered landscape of very specialized analysis scripts. With ETA, the user specifies the desired analysis method in a declarative style with a combination of graphical and traditional programming. Automatically selecting an appropriate algorithm, a just-in-time compiler combines these two inputs into an intermediate representation, which is then compiled into assembly code optimized for the target computer's architecture. This procedure optimizes for fast analysis of large time-tag files at the cost of some upfront compilation time, while still maintaining flexibility.

[1] https://timetag.github.io/.

2021 JINST 16 T08016

While several software solutions for the analysis of time-tagged data have been made available [23], ETA offers four key advantages:

1. ETA uses an optimized compiler to automatically provide short analysis times. This allows for correlation histograms to be viewed in real-time during the measurement. Direct feedback for alignment or data preview is a sought-after feature commonly missing for time-tagging since the data acquisition rate is too large to handle for standard software.

2. With our *Instrument Designer*, users can define the desired analysis method in a straight forward way by drawing state diagrams. Built-in functions and Python code can be executed whenever a new state is reached. This combination of state diagrams with custom code execution balances ease-of-use and flexibility when designing complex analysis methods.

3. We disentangle the experiment from vendor specific code by providing a unified user interface for the analysis of time-tagged data from all major time-to-digital converters.

4. ETA is open source and designed to grow with the help of the scientific community to rise to the challenges of the future.

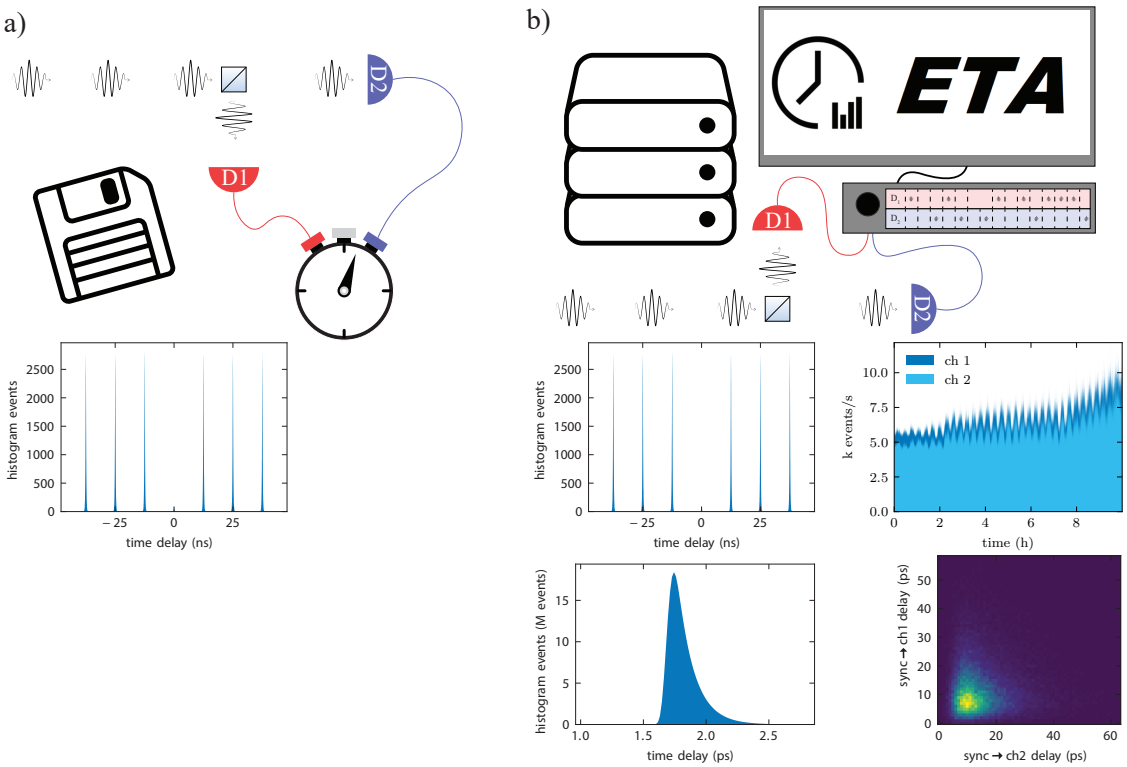


Figure 1. Configuration and results for time correlation measurements with (b) and without (a) time-tagging. (a) Correlating photons in time-to-analogue hardware yields the analyzed data directly with little storage space used. (b) When time-tagging photon arrival times, many different analysis methods can be applied to data from one experiment but a large amount of data is produced. Results depicted are: correlation, count rate, lifetime, and two-time correlation.

2021 JINST 16 T08016

We expect ETA to find uses in studies of single quantum emitters like atoms and molecules [24, 25], LIDAR [26], quantum entanglement [27–29] and fluorescence correlation spectroscopy measurements [30], as well as quantum key distribution protocols [31, 32] where data from remote detectors needs to be synchronized and correlated.

2 Software description

ETA combines a high-performance back end and a flexible, intuitive graphical user interface. The front end is used to design an analysis method in a graphical programming workflow. As illustrated in figure 2, it sends instructions to the back end, which compiles these instructions into optimized code and performs the analysis of the time-tagged data. The result is sent back to the front end for post-processing and displaying of the result. The user can describe how data is analyzed by creating a *Virtual Instrument*: after entering the *Instrument Designer* environment, a state diagram can be drawn where events, read from a time-tag file or even created on-the-fly, cause transitions among states. Upon arrival to a state or invocation of a specific transition, a user-defined action can be triggered. The specifics of the action are described in the coding panel on the right-hand side of the *Instrument Designer* using Python-like syntax. Multiple *Virtual Instruments* can be combined and used from within a *Script Panel*. There, additional data processing, analysis and plotting can be performed on the histogram calculated by the ETA back end. The programming language chosen for the *Script Panel* is Python. Standard functionality is provided for, amongst others, lifetime, correlation and count rate — all possible in real-time (see also section 3). In case a more specialized analysis method is desired, the user can build custom functionality by using provided functions or embedded code blocks for fully customized analysis.

2.1 Software architecture

Due to its division into front end and back end, ETA can be used in a multitude of ways, resulting in both cross-platform and cross-device compatibility. A WebSocket-based protocol is used for communication between front end and back end, allowing the computational power of the back end to be easily integrated into existing software. The front end is based on web-technologies to offer familiar aesthetics, displayed in a standalone software. A web-hosted version is available for mobile devices that support a browser. The back end installs as a Python package and provides a library interface, allowing integration with an individual Python workflow and easy installation across Microsoft Windows, Mac OS X, and Linux.

2.2 User interface (front end)

Main panel. The main panel, shown in figure 3, consists of a list composed of elements of one of three types:

- ***Parameters*** are strings that can be defined by the user on the *Main Panel* and interpreted as Boolean, integer, float or string in the *Instrument Designer* or *Script Panel*. In most included *Recipes* a *Parameter* is used to define the path to the time-tagged data.
- ***Virtual Instruments*** are at the heart of ETA. With the *Instrument Designer* instructions can be laid out inside the *Virtual Instruments* on how ETA analyzes time-tags. Each source of

2021 JINST 16 T08016

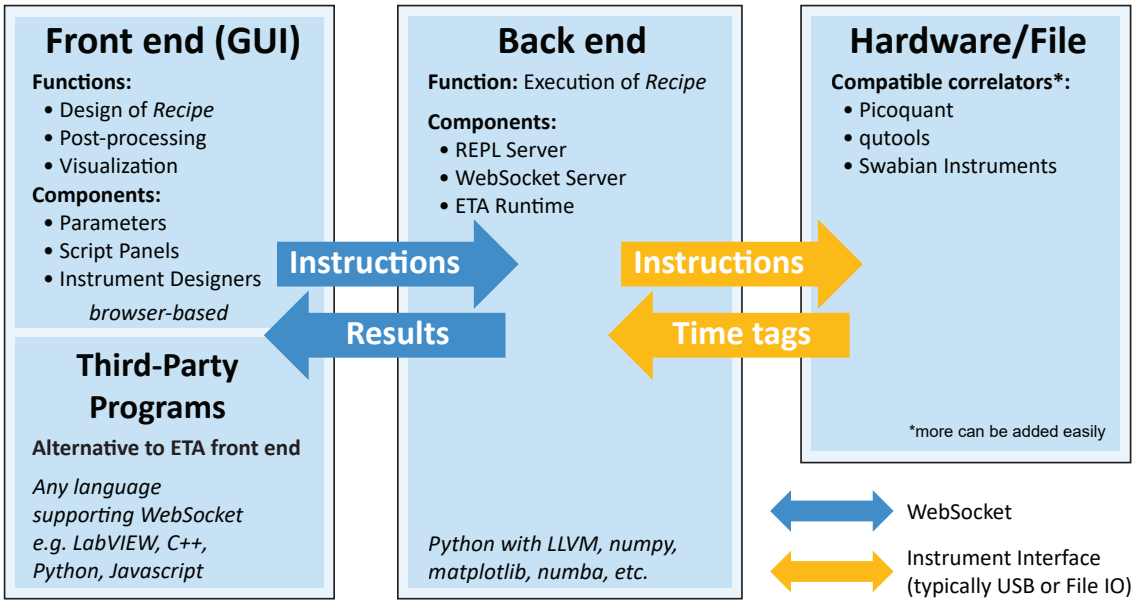


Figure 2. Software architecture of ETA. Time recording devices interact with the ETA back end via an interface like USB or via a saved file. The ETA back end receives its instructions from the ETA front end. Since the WebSocket protocol is used for the communication channel between back end and front end, a user-developed program can easily replace or extend the included front end, allowing for integration into an existing ecosystem.

time-tags, e.g. a measurement device, is represented as a *Virtual Instrument*. This allows the combination of time-tags from multiple devices. A delay line or a virtual beam splitter (see section 3) are further examples of *Virtual Instruments*.

- ***Script Panels*** are text editors with syntax highlighting for the Python language and are used to manage the analysis. Here the user decides which files to read and which *Virtual Instruments* to use. Also, it offers the possibility to do post-processing, secondary analysis, and visualization of the results returned by the *Virtual Instruments*.

The instrument designer. The *Instrument Designer* is divided into a state diagram on the left-hand side, e.g. figure 4a, and the *Actions* and *Tools* panel on the right-hand side , e.g. figure 4b. In the state diagram, blue circles represent states. They can be placed by double-clicking or by dragging out from an existing circle and can be named by double-clicking into the circle. There must be a single state where the analysis starts, which can be defined by selecting a state and pressing Shift+I (Initial). Connections between states, i.e. transitions (arrows), can be created by dragging out from an existing state onto another or onto itself. If a state is created by dragging out from an existing circle onto nothing, they will also be linked by a transition. Each transition must be labeled with all channel numbers that trigger this transition separated by commas. This can be done by double-clicking the transition. The *Actions and Tools* panel on the right-hand side defines what happens when a certain transition is triggered. This if-statement is represented by a description of the trigger followed by a colon and an indented block describing the Action. The trigger description

2021 JINST 16 T08016

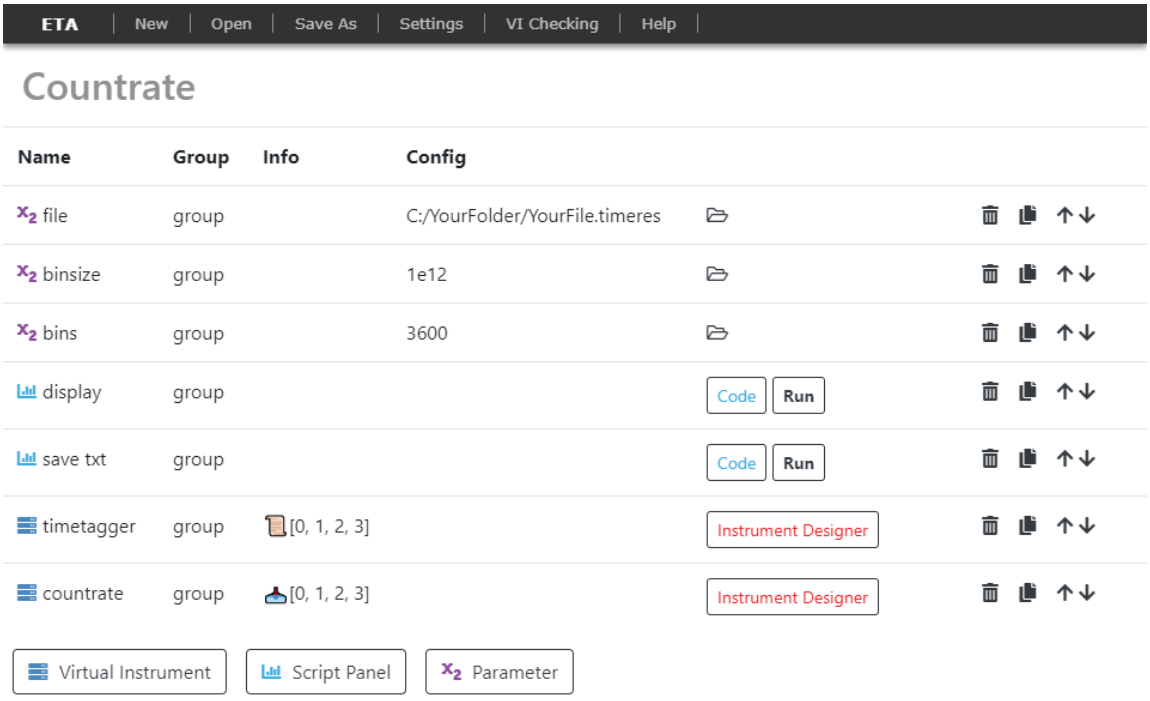


Figure 3. The *Main Panel* of ETA's front end with a simplified count rate *Recipe* loaded. The list of components shows three *Parameters*, two *Script Panels*, and two *Virtual Instruments*. A *Script Panel* can be opened by clicking the associated *Code* button. An *Instrument Designer* can be opened by clicking the corresponding button of the *Virtual Instrument*.

on the *Actions* and *Tools* side can be automatically created by clicking on a transition or a state and pressing Shift+T (Trigger) and works as detailed in table 1.

Table 1. Trigger declaration. Triggers are declared in the script panel of the *Instrument Designer*.

trigger declaration	meaning
A:	When state A is reached.
--2-->A:	When state A is reached via an event on channel 2.
B--1,2-->A:	When state A is reached from state B via an event on channel 1 or channel 2.

To perform *Actions* when a Trigger fires, often a *Tool* must be defined with which the *Action* can be performed. The most commonly used Tools are clocks and histograms which can be created by writing `CLOCK(clock_name)` and `HISTOGRAM(histogram_name, number_of_bins, width_of_bins)`, respectively. A clock can then be started and stopped with `clock_name.start()` and `clock_name.stop()`, while the recorded time difference can be entered into the histogram with `histogram_name.record(clock_name)` after the clock has been stopped. These *Actions* must be placed in the indented block of a trigger. Notably, the width of the bins can be set individually, thereby allowing e.g. logarithmic bins for capturing fast and slow processes in the same evaluation.

2021 JINST 16 T08016

2.3 Execution of analysis recipe (back end)

ETA is fast, efficient and compatible with a growing number of time-to-digital converters including PicoQuant, qutools, and Swabian Instruments. The compatibility with different file formats is provided through a flexible data loading layer. Below we highlight some of the algorithms and design choices responsible for ETA's efficiency and speed.

Algorithms. ETA automatically generates a highly optimized data processing program based on the analysis instructions defined in the recipe using both state diagram and *Tools* and *Actions* in the *Instrument Designer*. ETA accomplishes this by selecting and combining algorithms appropriately.

An N-way tournament sort algorithm [33] is used in the virtual channels, leveraging the fact that the time-tags are already pre-sorted in every channel, and most of the analysis methods are order-preserving. This means an event going into the delay line first comes out first, even if the absolute timing was changed inside the delay line. Based on this observation, using an N-way tournament sort can achieve a speedup of the time-tag analysis from a computational complexity $O(m\log(m))$ to $O(m\log(n))$, where m is the total number of events and n is the number of detector channels, compared to the Quicksort algorithm [34], a fast algorithm in the general case. This results in a speedup of $\frac{\log(m)}{\log(n)}$, which is typically 17x when doing correlation on a 1 GB time-tag file.

When performing correlation analysis, a ring buffer algorithm [35, 36] is used to reduce computational complexity from $O(m^2)$ to $O(mk)$, where m is the total number of events and k is the average number of two-channel correlation counts within the maximum time delay in the histogram. This results in a speedup of $\frac{m}{k}$, which can be several orders of magnitudes for large time-tag files with high event rates.

If the default choice of algorithm for implementing the *Tools* and *Actions* does not suit the user's needs, an embedded block of code, which can be written in Python, can also be used in the *Tools* and *Actions* panel. All of this code will then be combined and converted by Numba [37], a static Python compiler, into LLVM [38] code and afterwards into fast machine assembly.

Optimization. Unlike most of the existing data analysis tools, which become unwieldy and progressively slower as features are added, ETA uses a just-in-time compilation method. This allows ETA to compile only the algorithmic methods required for the current analysis, resulting in optimized native machine assembly code. Internally, ETA utilizes LLVM, a state-of-the-art assembly code generator and optimizer, to generate an intermediate representation for the analysis defined in the recipe. This takes into account the input time-tag file format and the variables defined in the *Virtual Instruments*. Variables are converted to constants before execution whenever possible to reduce the number of instructions at run-time. The intermediate representation is then translated using optimization tricks such as branch table generation and function-call elimination. This yields fast assembly code that runs directly on the target-CPU without an interpreter or virtual machine, resulting in performance similar to optimized C/C++ code for a specific type of analysis, while still maintaining flexibility via the *Instrument Designer*. Adding features does therefore not require a re-write of the program and a hard-to-manage code base with several similar analysis-specific functions that would have to be selected via computationally costly if-statements.

2021 JINST 16 T08016

3 Illustrative examples

3.1 Lifetime, start-stop, and correlation analysis

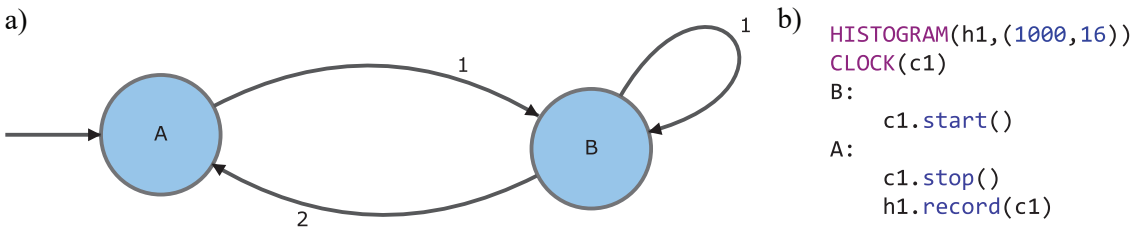


Figure 4. Lifetime analysis. (a) State diagram for lifetime analysis. (b) *Actions* and *Tools* for lifetime analysis.

Performing a lifetime analysis means sorting time differences between a synchronization event (sync) and the detection event, e.g. the arrival time of a photon. For a lifetime analysis, the state diagram has two states, e.g. A and B, where a transition from A to B starts a clock to measure the time difference and a transition from B to A stops this clock. The A → B transition (start) therefore occurs with an event on the sync channel and the B → A transition (stop) occurs with an event on the channel under investigation. If some photons are lost between source and detector, which is typically the case, several sync events can occur consecutively. To record only the shortest time differences, it is necessary to restart the clock on each sync event. We therefore, draw another transition from state B to itself, labeled with the sync channel number. We then trigger the Action `c1.start()` with the trigger `B:`, i.e. whenever we enter state B either from state A or from itself. And we trigger the *Actions* `c1.stop()` and `h1.record(c1)` with the trigger `A:`, i.e. whenever we enter state A, in this case only from state B. These are all the instructions specifying how ETA's back end analyzes the time-tagged data.

To load the measurement data we need to create a representation of the hardware on the *Main Panel* of the front end. Therefore, we create another *Virtual Instrument*, enter the *Instrument Designer* and specify the name of the source and number of channels that should be read from the file with `RFILE(timetagger_name,[0,1,2])`.

We then create a *Script Panel* which already includes the minimum example required to save the histogram to file:

```
import numpy as np
cut=eta.clips("C:\\Path_to_file\\File.timeres")
result= eta.run({"timetagger_name":cut})
histogram = result["h1"] # get list from result
np.savetxt("h1.txt",histogram) # save txt file
```

The whole analysis can then be executed by clicking the "Run"-button of the *Script Panel*. A more sophisticated version of a *Recipe* for lifetime analysis is included with the software.

Another prominent measurement, the second-order intensity time correlation function defined as $g^{(2)}(t) = \frac{\langle I(t)I(t+\tau)\rangle}{\langle I(t)^2\rangle}$ is essential in characterizing quantum sources: as an example, single photon emitters are identified with $g^{(2)}(0)$ values reaching well below 0.5 [39]. Auto-correlation measurements are also used in fluorescence correlation spectroscopy [40] and are often performed to identify the number of single-photon emitters, as well as diffusion and blinking times [41, 42].

2021 JINST 16 T08016

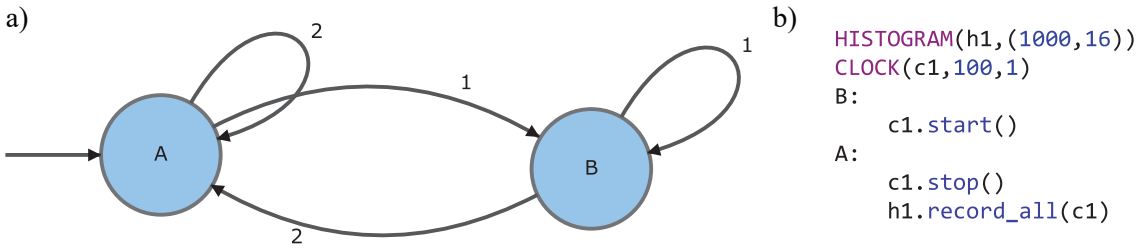


Figure 5. Correlation analysis. (a) State diagram for correlation analysis. (b) *Actions* and *Tools* for correlation analysis.

Formerly, simple start-stop measurements, that can be obtained with analog time to amplitude converters, were used to extract a reasonable approximation [43] that can yield the same result at $t = 0$, the value that is often of interest. In ETA, the start-stop *Recipe* can be easily made to simulate the behavior of the dedicated electronics in the old days, and it works similarly to the lifetime *Recipe*, by simply removing a reset transition.

With ETA, a real $g^{(2)}(t)$ measurement which contains more information than a start-stop measurement (see figure 6), can also be easily achieved with a correlation *Recipe*.

Since ETA can automatically produce an optimized code that is fast enough to perform full correlation, the only change required for the user is to specify in the *Instrument Designer*, that where time differences between all events, not only the neighbouring ones, have to be recorded into the histogram. This is done by allowing the clock to be started many times. Each started instance can be individually stopped when an event on the second channel is encountered. Since *Actions* have to take place when consecutive events occur on the same channel, both states A and B need a loop back to themselves. If we label them with the same channel numbers as the transitions pointing at them already, a second photon on the same channel will trigger another start of the clock in case of state B and a stop of the clock, as well as a recording to the histogram in case of state A. This will result in a correlation of the events on channel A with the events on channel B. A more sophisticated version of a *Recipe* for correlation analysis is included with the software.

Figure 6a shows an example of a raw event stream. Arrows indicate the recorded time intervals for positive (red) and negative (blue) time delay. As described before, a start-stop measurement will consider each event only once. When resetting the start, the last event in a row of consecutive events on the start channel will be used while the previous ones will be discarded. Since we can access the full event stream when time-tagging, to perform correlation, we can reuse any detection event to record time differences with all other events into a histogram. This is not possible with a simple start-stop type measurement. In figure 6b we illustrate this with a correlation where the stopping event is at most 6 time slots away from the starting event. Figure 6c shows a real-world example of data obtained from recording the arrival times of photons from a single quantum dot. The data in figure 6c has been normalized to the highest value in the correlation case for all three panels for easier comparison. A value of 1 does not represent the Poisson level in this case. While the result close to time delay 0 is similar between the start-stop evaluation with reset and the correlation evaluation, only correlation of all photons with each other provides accurate results for long time delays.

2021 JINST 16 T08016

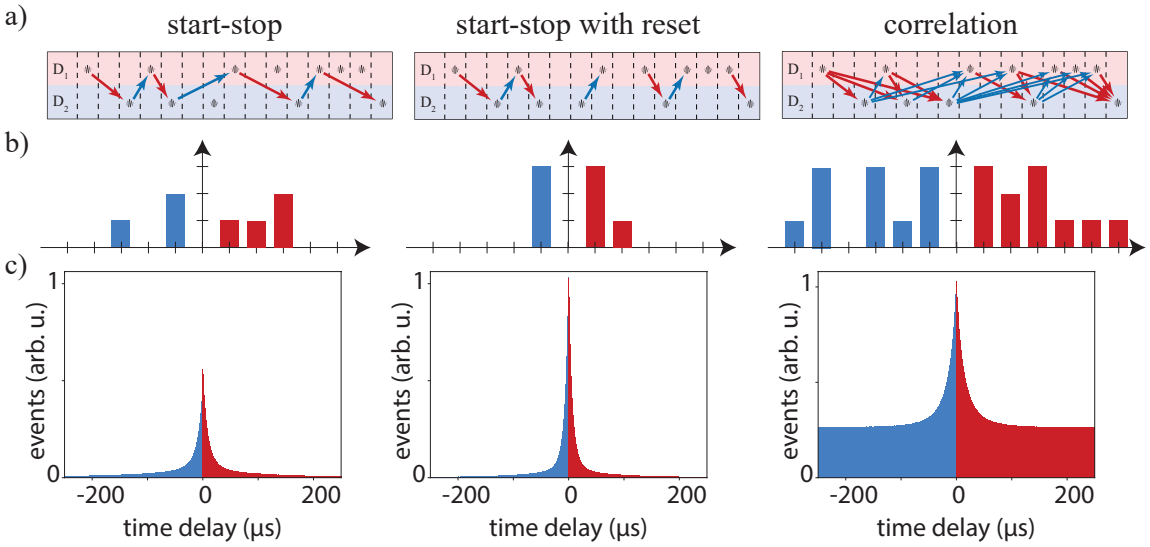


Figure 6. Comparison of different histogram calculations. Event streams (a), corresponding histograms (b), and exemplary data using fluorescence from a quantum dot (c) for, from left to right, start-stop analysis, start-stop with reset, and correlation with up to 6 neighbours.

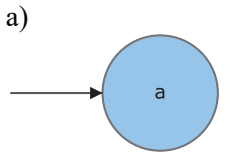
Real-time and multi-threading support. ETA allows processing new data while displaying and updating the result of the already evaluated data, which we call real-time analysis. Time-tag data can be streamed into ETA in segments for real-time analysis. Due to the nature of ETA, those features are implemented in the compiling stage and in a recipe-agnostic manner. This means real-time analysis can be enabled simply via the *Script Panel* in any type of ETA *Recipe*. It is provided with the built-in lifetime and correlation *Recipes*. The data can either be read from a file, while it is still being written or can be read-in directly from the time-to-digital converter. ETA will perform the analysis by automatically pausing and resuming as new data become available. In both cases, either the full time-tag file or only the analyzed data can be stored. This mode is often used for direct evaluation of a single-photon detector in an oscilloscope-like fashion. With this, a single-photon detector can, in many scenarios, replace a high-speed photo-diode, an important advantage when measuring very weak light intensities.

Previously stored data files can also be cut into smaller segments for faster parallel processing in a MapReduce [44] style method, in which the segments are analyzed individually into histograms (map stage), and aggregated with a user-specified method, usually a simple sum or concatenate (reduce stage). This can be useful to generate a quick preview of the analysis result at the cost of losing correlations between events across different segments.

3.2 Simulation

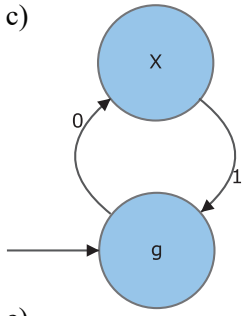
By using the `emit()` function in the *Instrument Designer* it is possible to create a custom time-tag file in memory or even on disk. To showcase this functionality, we create a stream of separated events suitable to simulate a pulsed $g^{(2)}(t)$ measurement with 100 % efficiency. The first *Virtual Instrument*, shown in figure 7a and 7 b, contains just one state with an initial-state marker, since that is the minimum requirement, and uses `emit(0, 0, 12496, 4.8E9)` to create an event on channel 1 with 0 ps delay every 12 496 ps, 4.8×10^9 times, resulting in 60 s of 80 MHz sync pulses. We then

2021 JINST 16 T08016



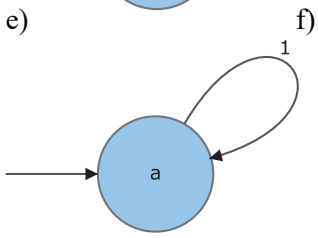
b)

```
emit(0,0,12496,4800000000)
```



d)

```
INTEGER(x_delay)
X:
    {
        binsize = 16
        x_delay_arr = ((np.random.exponential(250,1))/binsize))
        x_delay = round(x_delay_arr[0])*binsize
    }
    emit(1,x_delay)
```



f)

```
VFILE(2)
VFILE(3)
INTEGER(retchn)

a--1-->a:
    {
        options = np.asarray([2,3])
        retchn = np.random.choice(options)
    }
    emit(retchn, 0)
```

Figure 7. Simulation of time-correlation data. (a) State diagram for the generation of the sync with just the minimum requirements. (b) *Actions* and *Tools* panel for the generation of the sync with `emit(channel, delay,repetition_time,number_of_repetitions)`. (c) State diagram for the simulation of a single-photon emitter. (d) *Actions* and *Tools* for a delay randomly picked from an exponentially decaying distribution. (e) State diagram for a beam splitter triggered by the decay of the single-photon emitter. (f) *Actions* and *Tools* for the simulated beam splitter with a 50:50 choice of which output channel is used.

create a second *Virtual Instrument*, shown in figure 7c and 7d, with two states: a state called "g", representing the ground state and a state called "X" representing the excited state. On an event on channel 0, the ground state can be excited and upon arriving at "X" a delayed emission on channel 1 is triggered that will cause a return to the state "g". To generate this random delay, we make use of the embedded code block, where we can use all functions supported by the Numba compiler. By sampling from an exponential probability distribution, we can simulate the emission of a two-level system. `emit(1,x_delay)` is then called to create a virtual stream of single-photon-like events. In a third *Virtual Instrument* we pick up this stream with a transition from a single state looping to itself, by choosing the virtual channel 1 we just created. We then use another embedded code block to randomly choose between emitting on channel 2 or 3 with equal probability to mimic a 50:50 beam splitter. We can now correlate channel 2 and 3 as we did in section 3.1. A more sophisticated version of a quantum emitter simulation *Recipe* is included with the software. Figure 8 shows the result of this simulation, comparing correlation (top panel), start-stop analysis (middle panel) and start-stop analysis with reset of the clock upon consecutive events on the start channel (bottom panel). The start-stop method with reset only shows a single side-peak in each direction of the time delay since no event can be reused which would be necessary for a correlation analysis.

2021 JINST 16 T08016

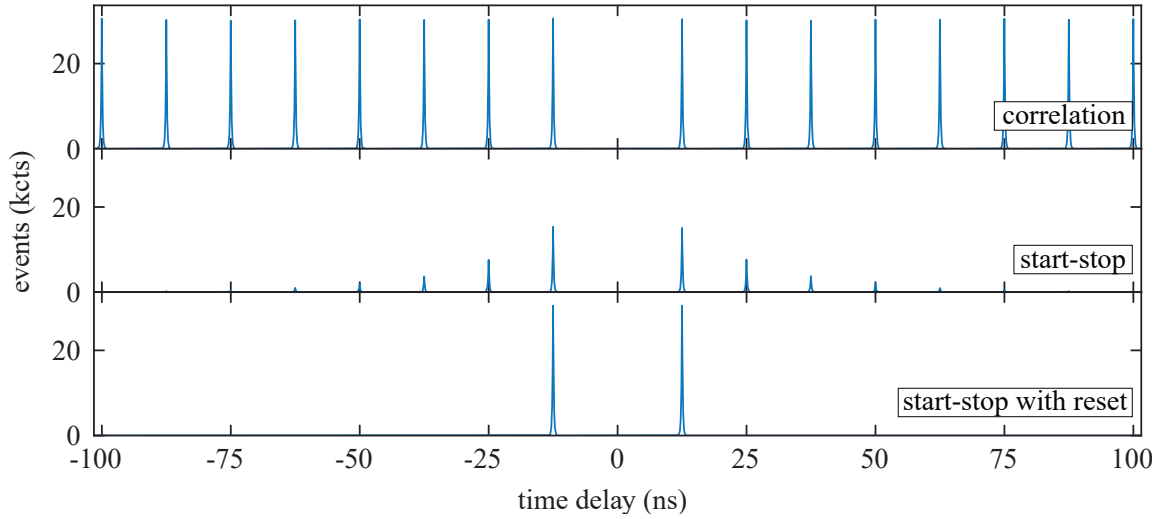


Figure 8. Simulated single-photon emission with 100 % efficiency in generation, collection, routing and detection. The top panel shows a full correlation, the middle panel shows a start-stop analysis and the bottom panel shows a start-stop analysis where the start is reset when consecutive photon events are registered on the same channel.

4 Conclusion

This software was developed with time-correlated single photon counting in mind. This technique is used in, among others, fluorescence microscopy and quantum optics, but is certainly not limited to these use cases. Analysis of time-tagged files instead of start-stop measurements allows for the extraction of as much information as possible from a single experiment, resulting in important time savings. Due to ETA's user-friendliness, it can also reduce the time spent on programming the analysis of recorded data. The program can perform novel analysis that are yet to be defined while remaining fast and robust. Large numbers of detectors, such as in Boson sampling [45] and ambitious linear optics quantum computation schemes [46] with associated large file sizes pose no problem for our software. The support of data from multiple time-taggers and from a multitude of vendors allows for use cases where the correlation of signals from remote sources, like in quantum key distribution, needs to be performed. Even simulation of time series data is possible due to the flexibility. A vast number of research fields could benefit from using our software due to time savings and unlocked potential.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 820423 (S2QUIP) and No. 899814 (Qurope), the Knut and Alice Wallenberg Foundation grant "Quantum Sensors", the European Research Council (307687 (NaQuOp)), the Joint China-Sweden Mobility programme (STINT), the Swedish Research Council (VR) through the VR grant for international recruitment of leading researchers (Ref: 2013-7152), and Q-LID Quantum Light Detectors (Ref: 2018-04251). K.D.J. acknowledges funding from the

2021 JINST 16 T08016

Swedish Research Council (VR) via the starting grant HyQRep (ref.: 2018-04812) and The Göran Gustafsson Foundation (SweTeQ).

Data sharing policy. Data sharing is not applicable to this article as no new data were created or analyzed in this study.

References

[1] M.B. Priestley, *Spectral analysis and time series*, Probability and Mathematical Statistics, Elsevier, London, U.K. (1982).

[2] I.E. Zadeh et al., *Efficient single-photon detection with* 7.7 *ps time resolution for photon-correlation measurements*, *ACS Photon.* **7** (2020) 1780.

[3] B. Korzh et al., *Demonstration of sub-*3 *ps temporal resolution with a superconducting nanowire single-photon detector*, *Nature Photon.* **14** (2020) 250.

[4] L.M. Bollinger and G.E. Thomas, *Measurement of the time dependence of scintillation intensity by a delayed-coincidence method*, *Rev. Sci. Instrum.* **32** (1961) 1044.

[5] P. Kapusta, M. Wahl and R. Erdmann eds., *Advanced photon counting: applications, methods, instrumentation*, Springer, Cham, Switzerland (2015).

[6] J. Shan and C.K. Toth, *Topographic laser ranging and scanning: principles and processing*, CRC Press, Taylor & Francis Group, (2018).

[7] M. Herrero-Collantes and J.C. Garcia-Escartin, *Quantum random number generators*, *Rev. Mod. Phys.* **89** (2017) 015004.

[8] C.J. Chunnilall, I.P. Degiovanni, S. Kück, I. Müller and A.G. Sinclair, *Metrology of single-photon sources and detectors: a review*, *Opt. Eng.* **53** (2014) 081910.

[9] C.A. Kocher and E.D. Commins, *Polarization correlation of photons emitted in an atomic cascade*, *Phys. Rev. Lett.* **18** (1967) 575.

[10] Y.H. Shih and C.O. Alley, *New type of Einstein-Podolsky-Rosen-Bohm experiment using pairs of light quanta produced by optical parametric down conversion*, *Phys. Rev. Lett.* **61** (1988) 2921.

[11] D.M. Greenberger, M.A. Horne and A. Zeilinger, *Going beyond Bell's theorem*, in *Bell's theorem, quantum theory and conceptions of the universe*, M. Kafatos ed., Springer, Dordrecht, The Netherlands (1989), pg. 69.

[12] H.J. Briegel and R. Raussendorf, *Persistent entanglement in arrays of interacting particles*, *Phys. Rev. Lett.* **86** (2001) 910.

[13] R.H. Brown and R.Q. Twiss, *Correlation between photons in two coherent beams of light*, *Nature* **177** (1956) 27.

[14] G.A. Steudle et al., *Measuring the quantum nature of light with a single source and a single detector*, *Phys. Rev. A* **86** (2012) 053814.

[15] G.W. Roberts and M. Ali-Bakhshian, *A brief introduction to time-to-digital and digital-to-time converters*, *IEEE Trans. Circuits Syst. II, Exp. Briefs2* **57** (2010) 153.

[16] M. Zopf et al., *Entanglement swapping with semiconductor-generated photons violates Bell's inequality*, *Phys. Rev. Lett.* **123** (2019) 160502.

2021 JINST 16 T08016

[17] F.B. Basset et al., *Entanglement swapping with photons generated on demand by a quantum dot*, *Phys. Rev. Lett.* **123** (2019) 160501.

[18] R. Ursin et al., *Entanglement-based quantum communication over* 144 *km*, *Nature Phys.* **3** (2007) 481.

[19] L.K. Shalm, D.R. Hamel, Z. Yan, C. Simon, K.J. Resch and T. Jennewein, *Three-photon energy-time entanglement*, *Nature Phys.* **9** (2012) 19.

[20] A. Delteil, Z. Sun, W.-B. Gao, E. Togan, S. Faelt and A. Imamoğlu, *Generation of heralded entanglement between distant hole spins*, *Nature Phys.* **12** (2015) 218.

[21] M. Reindl et al., *All-photonic quantum teleportation using on-demand solid-state quantum emitters*, *Science Adv.* **4** (2018) eaau1255.

[22] E. Schöll et al., *Resonance fluorescence of GaAs quantum dots with near-unity photon indistinguishability*, *Nano Lett.* **19** (2019) 2404.

[23] PicoQuant GmbH, *Time-resolved fluorescence software wiki*, https://perma.cc/FGT9-RD7T, September 2020.

[24] I. Aharonovich, D. Englund and M. Toth, *Solid-state single-photon emitters*, *Nature Photon.* **10** (2016) 631.

[25] W. Becker, *Fluorescence lifetime imaging — techniques and applications*, *J. Microscopy* **247** (2012) 119.

[26] G. Buller and A. Wallace, *Ranging and three-dimensional imaging using time-correlated single-photon counting and point-by-point acquisition*, *IEEE J. Sel. Top. Quant. Electron.* **13** (2007) 1006.

[27] S.J. Freedman and J.F. Clauser, *Experimental test of local hidden-variable theories*, *Phys. Rev. Lett.* **28** (1972) 938.

[28] K.D. Greve et al., *Quantum-dot spin-photon entanglement via frequency downconversion to telecom wavelength*, *Nature* **491** (2012) 421.

[29] W.B. Gao, P. Fallahi, E. Togan, J. Miguel-Sanchez and A. Imamoglu, *Observation of entanglement between a quantum dot spin and a single photon*, *Nature* **491** (2012) 426.

[30] M. Böhmer, F. Pampaloni, M. Wahl, H.-J. Rahn, R. Erdmann and J. Enderlein, *Time-resolved confocal scanning device for ultrasensitive fluorescence detection*, *Rev. Sci. Instrum.* **72** (2001) 4145.

[31] C.H. Bennett and G. Brassard, *Quantum cryptography: public key distribution and coin tossing*, in *Proceedings of IEEE international conference on computers, systems and signal processing*, volume 175, Bangalore, India, January 1984, pg. 8 [*Theor. Comput. Sci.* **560** (2014) 7].

[32] J. Yin et al., *Entanglement-based secure quantum cryptography over* 1, 120 *kilometres*, *Nature* **582** (2020) 501.

[33] D. Knuth, *The art of computer programming*, Addison-Wesley, U.S.A. (1968).

[34] C.A.R. Hoare, *Algorithm* 64*: quicksort*, *Commun. ACM* **4** (1961) 321.

[35] T. Bischof, *Photon correlation*, https://github.com/tsbischof/photon_correlation, February 2012.

[36] G. Ballesteros, R. Proux, C. Bonato and B.D. Gerardot, *readPTU: a python library to analyse time tagged time resolved data*, 2019 *JINST* **14** T06011 [arXiv:1903.07112].

[37] S.K. Lam, A. Pitrou and S. Seibert, *Numba: a LLVM-based python JIT compiler*, in *Proceedings of the second workshop on the LLVM compiler infrastructure in HPC — LLVM '15*, ACM press, (2015).

2021 JINST 16 T08016

[38] C. Lattner and V. Adve, *LLVM: a compilation framework for lifelong program analysis & Transformation*, in *Proceedings of the international symposium on code generation and optimization: feedback-directed and runtime optimization, CGO '*04, IEEE computer society, U.S.A., March 2004, pg. 75.

[39] H.J. Kimble, M. Dagenais and L. Mandel, *Photon antibunching in resonance fluorescence*, *Phys. Rev. Lett.* **39** (1977) 691.

[40] D. Magde, E. Elson and W.W. Webb, *Thermodynamic fluctuations in a reacting system-measurement by fluorescence correlation spectroscopy*, *Phys. Rev. Lett.* **29** (1972) 705.

[41] W.E. Moerner and D.P. Fromm, *Methods of single-molecule fluorescence spectroscopy and microscopy*, *Rev. Sci. Instrum.* **74** (2003) 3597.

[42] R.M. Dickson, A.B. Cubitt, R.Y. Tsien and W.E. Moerner, *On/off blinking and switching behaviour of single molecules of green fluorescent protein*, *Nature* **388** (1997) 355.

[43] F. Davidson and L. Mandel, *Photoelectric correlation measurements with time-to-amplitude converters*, *J. Appl. Phys.* **39** (1968) 62.

[44] J. Dean and S. Ghemawat, *MapReduce: simplified data processing on large clusters*, *Commun. ACM* **51** (2008) 107.

[45] H. Wang et al., *Boson sampling with* 20 *input photons and a* 60*-mode interferometer in a* 1014*-dimensional Hilbert space*, *Phys. Rev. Lett.* **123** (2019) 250503.

[46] E. Knill, R. Laflamme and G.J. Milburn, *A scheme for efficient quantum computation with linear optics*, *Nature* **409** (2001) 46.

2021 JINST 16 T08016