Efficient and Versatile Toolbox for Analysis of Time-Tagged Measurements

Z. Lin,[a,b,i] **L. Schweickert,**[a,i,ii] **S. Gyger,**[a] **K. D. Jöns,**[a,iii] **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, the 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 events. It is specifically optimized for, but not limited to, time correlation extraction from time-stamped events. A combination of state diagrams and simple code snippets allows for analysis of arbitrary complexity while keeping computational efficiency high. Our tool allows for flexible extraction of correlation from time-tagged data beyond start-stop measurements that were traditionally used.

Keywords: Data processing methods, Detector control systems, Analysis and statistical methods, Data reduction methods, Simulation methods and programs

[i] Authors contributed equally.

[ii] Corresponding author.

[iii] Current affiliation: Department of Physics, Paderborn University, 33098 Paderborn, Germany

Contents

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 picosecond 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 a 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 (Figure 1 a)). 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 (Figure 1 b), 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.

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. Rudimentary software offerings from manufacturers of time-to-digital converters 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. Time-tagged files, on the other hand, 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 software-defined analysis 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),[i] 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.

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 on-the-fly compiler to automatically provide short analysis times.

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

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.

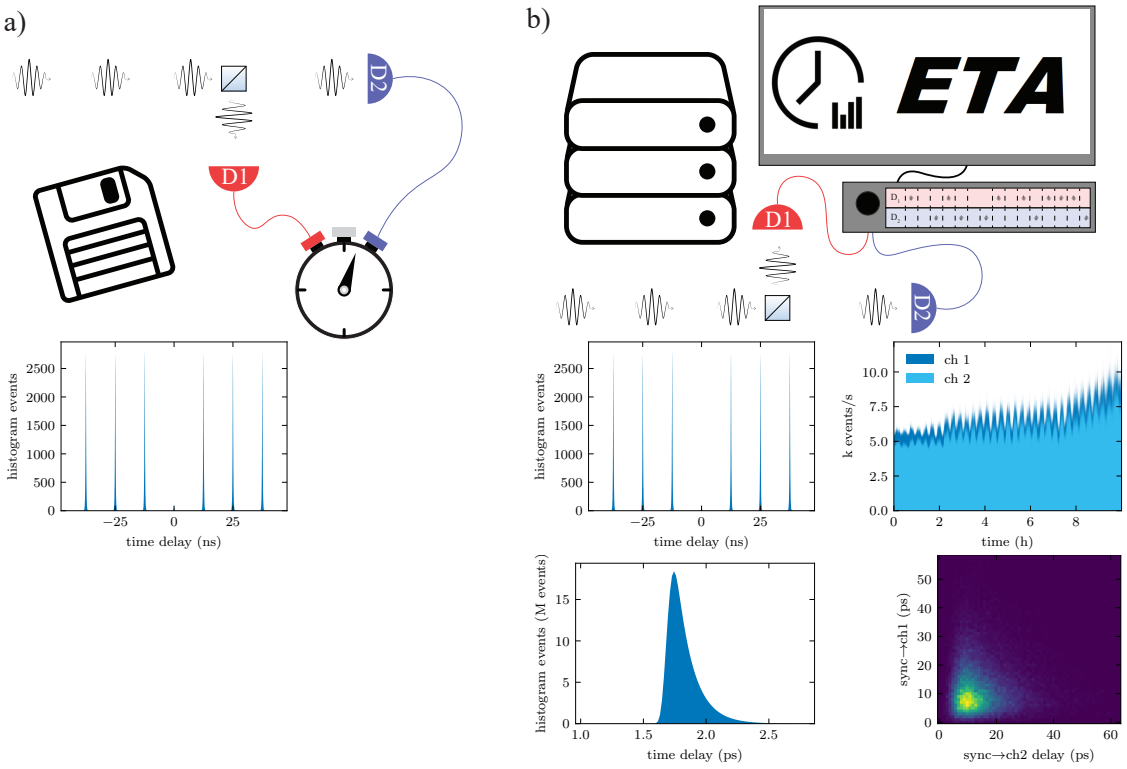


Figure 1. Configuration and results for time correlation measurements with (b)) and without (a)) time-tagging. a) Correlating photons in time-to-analogue hardware yields the analyzed data directly with little storage space used. (b) When time-tagging photon arrival times, many different analysis methods 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.

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 flexibility with an intuitive user interface in a graphical programming workflow. 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 using Python-like syntax. Multiple *Virtual Instruments* can be combined and used from within the *Script Panel*. There, additional data processing, analysis and plotting can be performed on the histogram calculated by the ETA backend. 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 Tutorial section). In case a more specialized analysis method is desired, a sandbox approach lets the user build custom functionality by using the internal functions or embedded Python for fully customized analysis.

2.1 Software Architecture

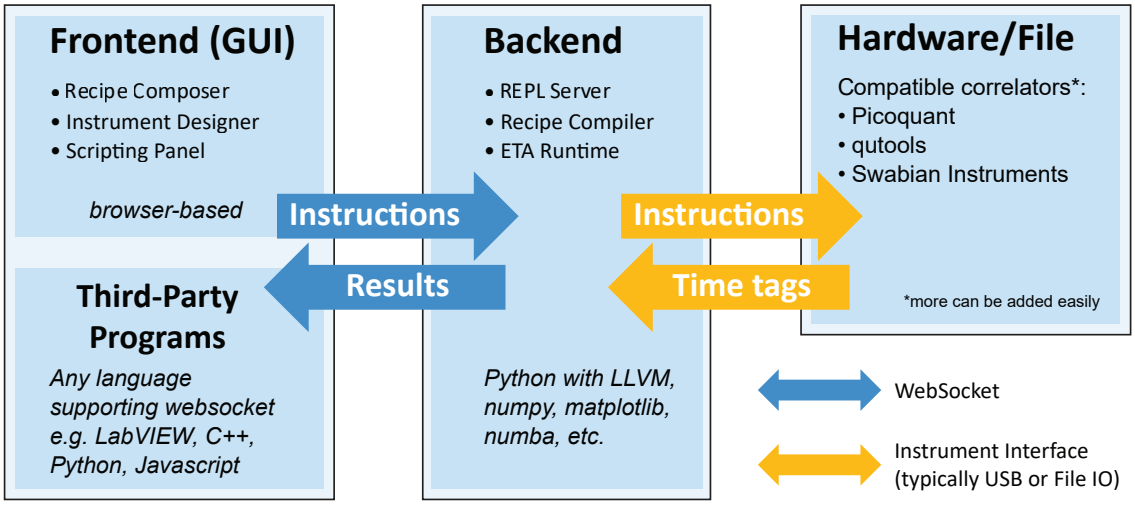


Figure 2. Software architecture of ETA. Time recording devices interact with the ETA backend via an interface like USB or via a saved file. The ETA backend receives its instructions from the ETA frontend. Since the WebSocket protocol is used for the communication channel between backend and frontend,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

Due to its division into frontend and backend, 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 frontend and backend, allowing the computational power of the backend to be easily integrated into existing software. The frontend 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 backend 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 (Frontend)

Main Panel

The main panel consists of a list composed of elements of one of three types:

- *Parameters*
- *Virtual Instruments*
- *Script Panels*

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*.

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 time-tags, e.g. a measurement device, is represented as a *Virtual Instrument*. This allows the combination of time-tags from multiple devices.

Script Panels 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. Fig. 3a)) and the *Actions* and *Tools* panel on the right-hand side (e.g. Fig. 3b)). 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 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.

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.

Table 1. Trigger declaration. Triggers are declared in the 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.

2.3 Analysis optimization (backend)

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.

Meta-programming

ETA automatically generates a highly optimized data processing routine based on the analysis instructions defined using both state diagram and *Tools* and *Actions* in the *Instrument Designer*. ETA accomplishes this by selecting the ideal algorithms and combining them 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, embedded 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.

Just-in-time compilation

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 from the processing routine. This takes into account the target-CPU architecture, 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, function-call elimination, memory alignment and instruction. 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.

Real-time and multi-threading support

Time-tag data can be streamed into ETA in chunks for real-time analysis. Due to the meta-programming and just-in-time compiling nature of ETA, those features are implemented in the compiling stage and a recipe-agnostic manner. This means an existing recipe can use real-time processing by enabling it in the *Code Panel* (Python) and ETA will perform the analysis by automatically pausing and resuming as new data become available. Data files can also be cut into smaller chunks for faster parallel processing in a MapReduce [39] style method, in which the chunks 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. However, correlations between events of different segments are then lost.

3 Illustrative examples

3.1 Lifetime, Start-stop, and Correlation analysis

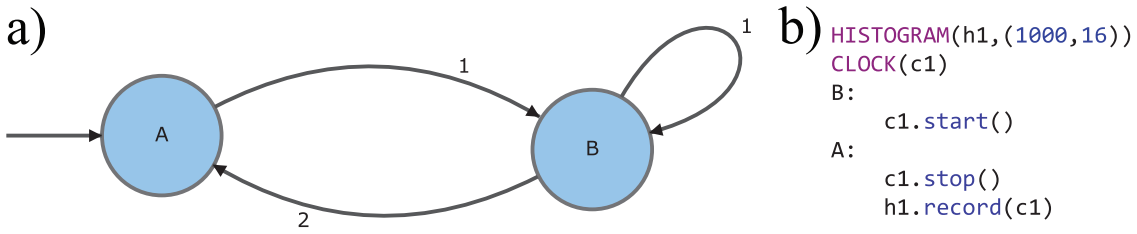


Figure 3. Lifetime analysis. a) State diagram for lifetime analysis. b) *Actions* and *Tools* for lifetime analysis.

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 $\rightarrow$ B transition (start) therefore occurs with an event on the sync channel and the B $\rightarrow$ 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 backend 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 frontend. 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.

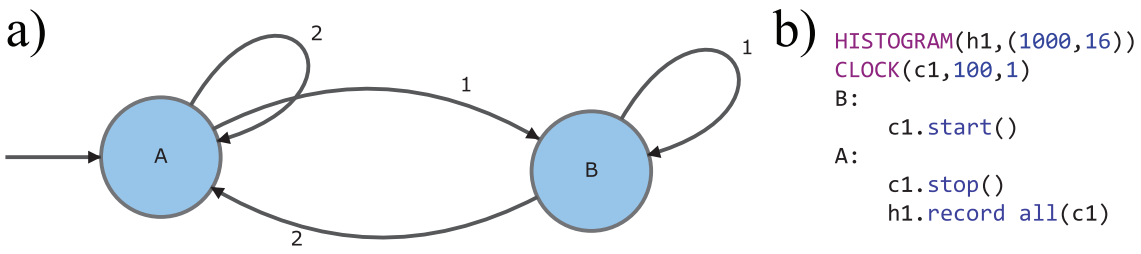


Figure 4. Correlation analysis. a) State diagram for correlation analysis. b) *Actions* and *Tools* for correlation analysis.

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 [40]. Auto-correlation measurements are also used in fluorescence correlation spectroscopy [41] and are often performed to identify the number of single-photon emitters, as well as diffusion and blinking times [42, 43].

Formerly, simple start-stop measurements, that can be obtained with analog time to amplitude converters, were used to extract a reasonable approximation [44] 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 curve.

With ETA, a real $g^{(2)}(t)$ measurement which contains more information than a start-stop measurement (see Fig. 5), 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*, 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 5 a) 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 5 b) we illustrate this with a correlation where the stopping event is at most 6 time slots away from the starting event. Figure 5 c) shows a real-world example of data obtained from recording the arrival times of photons from a single quantum dot. The data in c) 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.

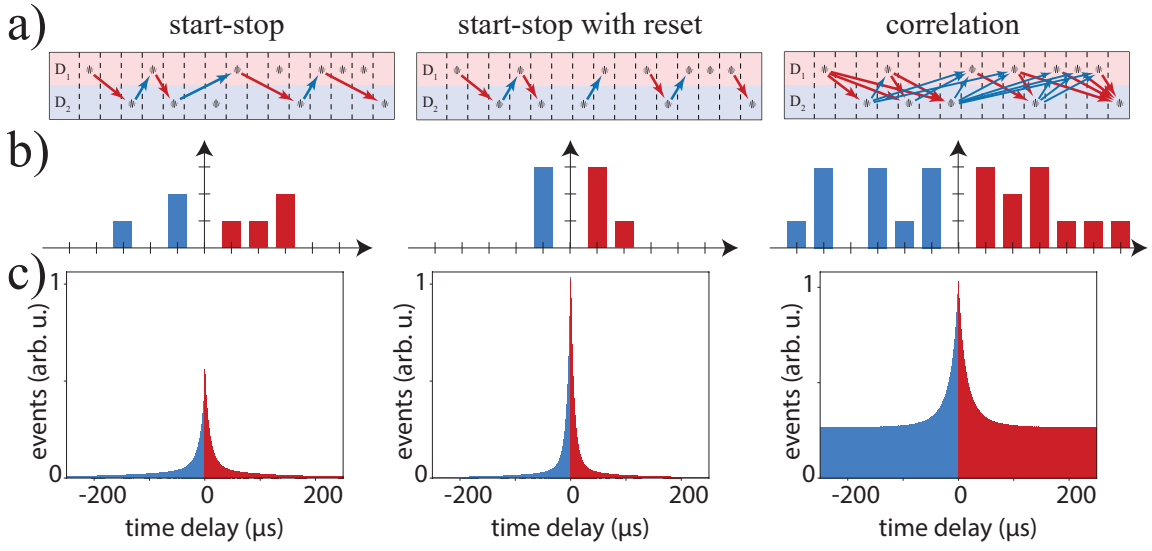
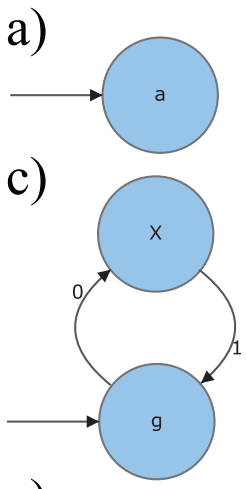


Figure 5. Comparison of different histogram calculations. Event streams a), corresponding histograms b), and exemplary data using fluorescence from a quantum dot c) for (from left to right) start-stop analysis, start-stop with reset, and correlation with up to 6 neighbours.

3.2 Realtime analysis

ETA allows processing new data while displaying and updating the result of the already evaluated data, which we call realtime analysis. This can be added simply via the *Script Panel* to any type of ETA *Recipe* and is provided with 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. In both cases, either the full time-tag file or only the analyzed data can be stored. We use this mode for direct evaluation of a single-photon detector in an oscilloscope-like fashion. With this, a single-photon detector can replace a high-speed photodiode, an important advantage when measuring very weak light intensities.

3.3 Simulation



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 6. 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.

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 6 a) and 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 create a second *Virtual Instrument*, shown in Figure 6 c) and d), 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 Python environment, where we can use all functions supported by the Numba interpreter. 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 Python 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 7 shows the result of this simulation, comparing correlation (top panel),

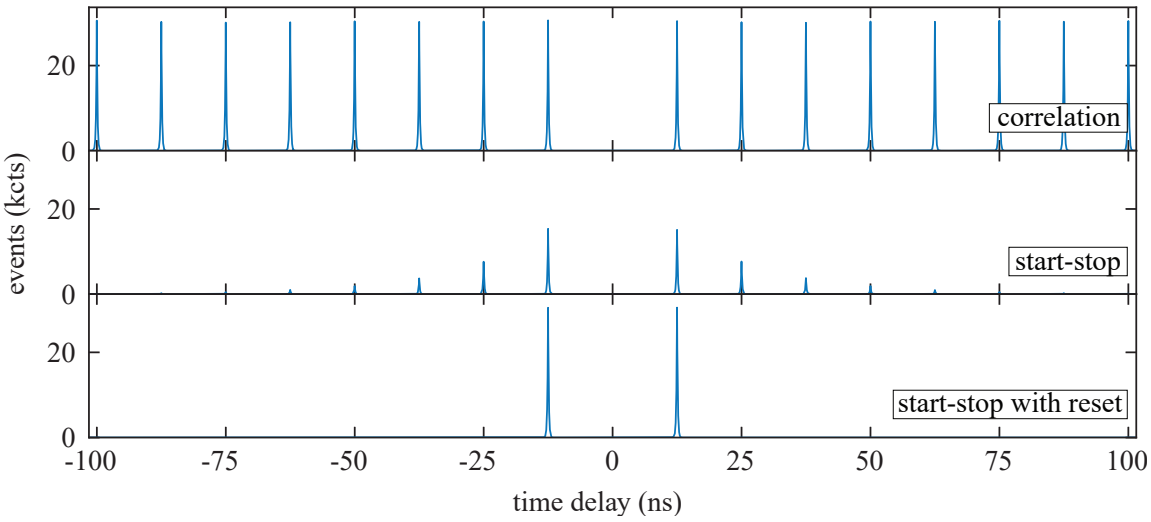


Figure 7. Simulated single-photon emission with 100 % efficiency in generation, collection, routing and detection. The top panel shows a full correlation, the middle panel shows a start-stop analysis and the bottom panel shows a start-stop analysis where the start is reset when consecutive photon events are registered on the same channel.

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.

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 extract novel correlations 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 flexible programming with embedded Python. 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 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, October 1982. ISBN 978-0-12-564922-3.

[2] Iman Esmaeil Zadeh, Johannes W. N. Los, Ronan B. M. Gourgues, Jin Chang, Ali W. Elshaari, Julien Romain Zichi, Yuri J. van Staaden, Jeroen P. E. Swens, Nima Kalhor, Antonio Guardiani, Yun Meng, Kai Zou, Sergiy Dobrovolskiy, Andreas W. Fognini, Dennis R. Schaart, Dan Dalacu, Philip J. Poole, Michael E. Reimer, Xiaolong Hu, Silvania F. Pereira, Val Zwiller, and Sander N. Dorenbos. Efficient Single-Photon Detection with 7.7 ps Time Resolution for Photon-Correlation Measurements. *ACS Photonics*, 7(7):1780–1787, July 2020. doi: 10.1021/acsphotonics.0c00433.

[3] Boris Korzh, Qing-Yuan Zhao, Jason P. Allmaras, Simone Frasca, Travis M. Autry, Eric A. Bersin, Andrew D. Beyer, Ryan M. Briggs, Bruce Bumble, Marco Colangelo, Garrison M. Crouch, Andrew E. Dane, Thomas Gerrits, Adriana E. Lita, Francesco Marsili, Galan Moody, Cristián Peña, Edward Ramirez, Jake D. Rezac, Neil Sinclair, Martin J. Stevens, Angel E. Velasco, Varun B. Verma, Emma E. Wollman, Si Xie, Di Zhu, Paul D. Hale, Maria Spiropulu, Kevin L. Silverman, Richard P. Mirin, Sae Woo Nam, Alexander G. Kozorezov, Matthew D. Shaw, and Karl K. Berggren. Demonstration of sub-3 ps temporal resolution with a superconducting nanowire single-photon detector. *Nature Photonics*, 14(4):250–255, April 2020. doi: 10.1038/s41566-020-0589-x.

[4] L. M. Bollinger and G. E. Thomas. Measurement of the Time Dependence of Scintillation Intensity by a Delayed-Coincidence Method. *Review of Scientific Instruments*, 32(9):1044–1050, September 1961. doi: 10.1063/1.1717610.

[5] Peter Kapusta, Michael Wahl, and Rainer Erdmann, editors. *Advanced Photon Counting: Applications, Methods, Instrumentation*, volume 15 of *Springer Series on Fluorescence*. Springer International Publishing, Cham, 2015. ISBN 978-3-319-15635-4 978-3-319-15636-1. doi: 10.1007/978-3-319-15636-1.

[6] Jie Shan and Charles K Toth. *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, Taylor & Francis Group, 2018. ISBN 978-1-315-15438-1.

[7] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. Quantum random number generators. *Reviews of Modern Physics*, 89(1):015004, February 2017. doi: 10.1103/RevModPhys.89.015004.

[8] Christopher J. Chunnilall, Ivo Pietro Degiovanni, Stefan Kück, Ingmar Müller, and Alastair G. Sinclair. Metrology of single-photon sources and detectors: A review. *Optical Engineering*, 53(8): 081910, July 2014. doi: 10.1117/1.OE.53.8.081910.

[9] Carl A. Kocher and Eugene D. Commins. Polarization Correlation of Photons Emitted in an Atomic Cascade. *Physical Review Letters*, 18(15):575–577, April 1967. doi: 10.1103/PhysRevLett.18.575.

[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. *Physical Review Letters*, 61(26): 2921–2924, December 1988. doi: 10.1103/PhysRevLett.61.2921.

[11] Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. Going Beyond Bell's Theorem. In Menas Kafatos, editor, *Bell's Theorem, Quantum Theory and Conceptions of the Universe*, Fundamental Theories of Physics, pages 69–72. Springer Netherlands, Dordrecht, 1989. ISBN 978-94-017-0849-4. doi: 10.1007/978-94-017-0849-4_10.

[12] Hans J. Briegel and Robert Raussendorf. Persistent Entanglement in Arrays of Interacting Particles. *Physical Review Letters*, 86(5):910–913, January 2001. doi: 10.1103/PhysRevLett.86.910.

[13] R. Hanbury Brown and R. Q. Twiss. Correlation between Photons in two Coherent Beams of Light. *Nature*, 177(4497):27–29, January 1956. doi: 10.1038/177027a0.

[14] Gesine A. Steudle, Stefan Schietinger, David Höckel, Sander N. Dorenbos, Iman E. Zadeh, Valery Zwiller, and Oliver Benson. Measuring the quantum nature of light with a single source and a single detector. *Physical Review A*, 86(5):053814, November 2012. doi: 10.1103/PhysRevA.86.053814.

[15] Gordon W. Roberts and Mohammad Ali-Bakhshian. A Brief Introduction to Time-to-Digital and Digital-to-Time Converters. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(3): 153–157, March 2010. doi: 10.1109/TCSII.2010.2043382.

[16] Michael Zopf, Robert Keil, Yan Chen, Jingzhong Yang, Disheng Chen, Fei Ding, and Oliver G. Schmidt. Entanglement Swapping with Semiconductor-Generated Photons Violates Bell's Inequality. *Physical Review Letters*, 123(16):160502, October 2019. doi: 10.1103/PhysRevLett.123.160502.

[17] F. Basso Basset, M. B. Rota, C. Schimpf, D. Tedeschi, K. D. Zeuner, S. F. Covre da Silva, M. Reindl, V. Zwiller, K. D. Jöns, A. Rastelli, and R. Trotta. Entanglement Swapping with Photons Generated on Demand by a Quantum Dot. *Physical Review Letters*, 123(16):160501, October 2019. doi: 10.1103/PhysRevLett.123.160501.

[18] R. Ursin, F. Tiefenbacher, T. Schmitt-Manderbach, H. Weier, T. Scheidl, M. Lindenthal, B. Blauensteiner, T. Jennewein, J. Perdigues, P. Trojek, B. Ömer, M. Fürst, M. Meyenburg, J. Rarity,

Z. Sodnik, C. Barbieri, H. Weinfurter, and A. Zeilinger. Entanglement-based quantum communication over 144 km. *Nature Physics*, 3(7):481–486, July 2007. doi: 10.1038/nphys629.

[19] L. K. Shalm, D. R. Hamel, Z. Yan, C. Simon, K. J. Resch, and T. Jennewein. Three-photon energy–time entanglement. *Nature Physics*, 9(1):19–22, January 2013. doi: 10.1038/nphys2492.

[20] Aymeric Delteil, Zhe Sun, Wei-bo Gao, Emre Togan, Stefan Faelt, and Ataç Imamoğlu. Generation of heralded entanglement between distant hole spins. *Nature Physics*, 12(3):218–223, March 2016. doi: 10.1038/nphys3605.

[21] Marcus Reindl, Daniel Huber, Christian Schimpf, Saimon F. Covre da Silva, Michele B. Rota, Huiying Huang, Val Zwiller, Klaus D. Jöns, Armando Rastelli, and Rinaldo Trotta. All-photonic quantum teleportation using on-demand solid-state quantum emitters. *Science Advances*, 4(12): eaau1255, December 2018. doi: 10.1126/sciadv.aau1255.

[22] Eva Schöll, Lukas Hanschke, Lucas Schweickert, Katharina D. Zeuner, Marcus Reindl, Saimon Filipe Covre da Silva, Thomas Lettner, Rinaldo Trotta, Jonathan J. Finley, Kai Müller, Armando Rastelli, Val Zwiller, and Klaus D. Jöns. Resonance Fluorescence of GaAs Quantum Dots with Near-Unity Photon Indistinguishability. *Nano Letters*, 19(4):2404–2410, April 2019. doi: 10.1021/acs.nanolett.8b05132.

[23] PicoQuant GmbH. Software - Time-Resolved Fluoresence Wiki. https://perma.cc/FGT9-RD7T, September 2020.

[24] Igor Aharonovich, Dirk Englund, and Milos Toth. Solid-state single-photon emitters. *Nature Photonics*, 10(10):631–641, October 2016. doi: 10.1038/nphoton.2016.186.

[25] W. Becker. Fluorescence lifetime imaging – techniques and applications. *Journal of Microscopy*, 247 (2):119–136, 2012. doi: 10.1111/j.1365-2818.2012.03618.x.

[26] Gerald Buller and Andrew Wallace. Ranging and Three-Dimensional Imaging Using Time-Correlated Single-Photon Counting and Point-by-Point Acquisition. *IEEE Journal of Selected Topics in Quantum Electronics*, 13(4):1006–1015, July 2007. doi: 10.1109/JSTQE.2007.902850.

[27] Stuart J. Freedman and John F. Clauser. Experimental Test of Local Hidden-Variable Theories. *Physical Review Letters*, 28(14):938–941, April 1972. doi: 10.1103/PhysRevLett.28.938.

[28] Kristiaan De Greve, Leo Yu, Peter L. McMahon, Jason S. Pelc, Chandra M. Natarajan, Na Young Kim, Eisuke Abe, Sebastian Maier, Christian Schneider, Martin Kamp, Sven Höfling, Robert H. Hadfield, Alfred Forchel, M. M. Fejer, and Yoshihisa Yamamoto. Quantum-dot spin–photon entanglement via frequency downconversion to telecom wavelength. *Nature*, 491(7424):421–425, November 2012. doi: 10.1038/nature11577.

[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(7424):426–430, November 2012. doi: 10.1038/nature11573.

[30] Martin Böhmer, Francesco Pampaloni, Michael Wahl, Hans-Jürgen Rahn, Rainer Erdmann, and Jörg Enderlein. Time-resolved confocal scanning device for ultrasensitive fluorescence detection. *Review of Scientific Instruments*, 72(11):4145–4152, November 2001. doi: 10.1063/1.1406926.

[31] Charles Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, volume 175, page 8, Bangelore, India, January 1984. doi: 10.1016/j.tcs.2011.08.039.

[32] Juan Yin, Yu-Huai Li, Sheng-Kai Liao, Meng Yang, Yuan Cao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Shuang-Lin Li, Rong Shu, Yong-Mei Huang, Lei Deng, Li Li, Qiang Zhang, Nai-Le Liu, Yu-Ao Chen, Chao-Yang Lu, Xiang-Bin Wang, Feihu Xu, Jian-Yu Wang, Cheng-Zhi

Peng, Artur K. Ekert, and Jian-Wei Pan. Entanglement-based secure quantum cryptography over 1,120 kilometres. *Nature*, 582(7813):501–505, June 2020. doi: 10.1038/s41586-020-2401-y.

[33] Donald Knuth. *The Art of Computer Programming*. Addison-Wesley, 1968. ISBN 0-201-03801-3.

[34] C. A. R. Hoare. Algorithm 64: Quicksort. *Communications of the ACM*, 4(7):321, July 1961. doi: 10.1145/366622.366644.

[35] Thomas Bischof. Photon Correlation, February 2012.

[36] G. C. Ballesteros, R. Proux, C. Bonato, and B. D. Gerardot. readPTU: A python library to analyse time tagged time resolved data. *Journal of Instrumentation*, 14(06):T06011–T06011, June 2019. doi: 10.1088/1748-0221/14/06/T06011.

[37] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15*, pages 1–6, Austin, Texas, 2015. ACM Press. ISBN 978-1-4503-4005-2. doi: 10.1145/2833157.2833162.

[38] Chris Lattner and Vikram 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, page 75, USA, March 2004. IEEE Computer Society. ISBN 978-0-7695-2102-2.

[39] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, January 2008. doi: 10.1145/1327452.1327492.

[40] H. J. Kimble, M. Dagenais, and L. Mandel. Photon Antibunching in Resonance Fluorescence. *Physical Review Letters*, 39(11):691–695, September 1977. doi: 10.1103/PhysRevLett.39.691.

[41] Douglas Magde, Elliot Elson, and W. W. Webb. Thermodynamic Fluctuations in a Reacting System—Measurement by Fluorescence Correlation Spectroscopy. *Physical Review Letters*, 29(11): 705–708, September 1972. doi: 10.1103/PhysRevLett.29.705.

[42] W. E. Moerner and David P. Fromm. Methods of single-molecule fluorescence spectroscopy and microscopy. *Review of Scientific Instruments*, 74(8):3597–3619, July 2003. doi: 10.1063/1.1589587.

[43] Robert M. Dickson, Andrew B. Cubitt, Roger Y. Tsien, and W. E. Moerner. On/off blinking and switching behaviour of single molecules of green fluorescent protein. *Nature*, 388(6640):355–358, July 1997. doi: 10.1038/41048.

[44] F. Davidson and L. Mandel. Photoelectric Correlation Measurements with Time-to-Amplitude Converters. *Journal of Applied Physics*, 39(1):62–66, January 1968. doi: 10.1063/1.1655781.

[45] Hui Wang, Jian Qin, Xing Ding, Ming-Cheng Chen, Si Chen, Xiang You, Yu-Ming He, Xiao Jiang, L. You, Z. Wang, C. Schneider, Jelmer J. Renema, Sven Höfling, Chao-Yang Lu, and Jian-Wei Pan. Boson sampling with 20 input photons and a 60-mode interferometer in a 10^{14}–Dimensional hilbert space. *Physical Review Letters*, 123(25):250503, December 2019. doi: 10.1103/PhysRevLett.123.250503.

[46] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, January 2001. doi: 10.1038/35051009.