EPJ Web of Conferences **245**, 02018 (2020) https://doi.org/10.1051/epjconf/202024502018
CHEP 2019

The Heavy Photon Search (HPS) Software Environment

Norman Graf for the HPS Collaboration[1,*]

[1]SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA, 94025, USA

Abstract. The Heavy Photon Search (HPS) is an experiment at the Thomas Jefferson National Accelerator Facility (JLab) designed to search for a hidden sector photon (A') in fixed-target electro-production. It uses a silicon microstrip tracking and vertexing detector placed inside a dipole magnet to measure charged particle trajectories and a fast lead-tungstate crystal calorimeter located just downstream of the magnet to provide a trigger and to identify electromagnetic showers. The HPS experiment uses both invariant mass and secondary vertex signatures to search for the A'. The experimental collaboration is small and quite heterogeneous: it is composed of members of the nuclear physics as well as particle physics communities, from universities and national labs from around the US and Europe. Enabling such a disparate group to concentrate on the physics aspects of the experiment required that the software be easy to install and use, and having such limited manpower meant that existing solutions had to be exploited.
HPS has successfully completed two engineering runs and completed its first physics run in the summer of 2019. We begin with an overview of the physics goals of the experiment followed by a short description of the detector design. We then describe the software tools used to design the detector layout and simulate the expected detector performance. Event reconstruction involving track, cluster and vertex finding and fitting for both simulated and real data was, to first order, adopted from existing software originally developed for Linear Collider studies. Bringing it all together into a cohesive whole involved the use of multiple software solutions with common interfaces.

1 Introduction

There is strong observational evidence for the existence of Dark Matter but its specific nature continues to elude us. The HPS experiment searches for a so-called Heavy Photon (*aka* dark photon or A') which is one well-motivated portal for Dark Matter interactions with the Standard Model[1][2]. Kinetic mixing between the Standard Model photon and the dark photon would induce weak coupling to electric charge. If dark photons do couple to electric charge in this way, they will be produced through a process analogous to bremsstrahlung off heavy targets, subsequently decaying to lepton pairs if above the mass threshold for such production. The kinematics of this process are very different from bremsstrahlung: production is sharply forward-peaked as the A' takes most of the beam energy, as shown in Figure 1.

The HPS experiment was designed to make use of such a production mechanism to search for a heavy photon decaying into electron-positron pairs using two methods. The first resonance search looks for an excess in the invariant mass distribution of electron-positron pairs

*e-mail: Norman.Graf@slac.stanford.edu

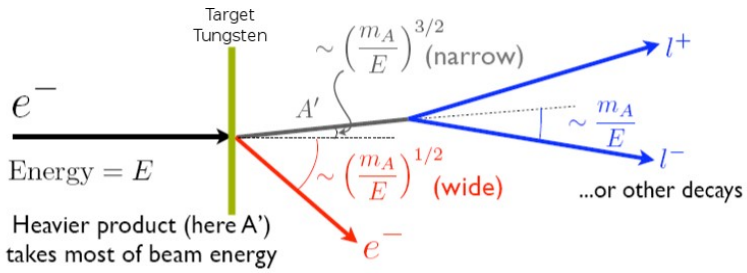

Figure 1. Characteristic production angles for the A' and its leptonic decay products.

above the large QED background. The other searches for displaced vertices resulting from long-lived A' decays. Both analyses require fast, robust, high-precision detector elements placed extremely close to a high-intensity electron beam to not only accumulate high statistics, but also to precisely reconstruct the decay products in order to detect and measure any A' decays above an enormous Standard Model background.

2 Detector Response Simulation

The short timescale for initial development of the experiment demanded quick turnaround for the detector design. In order to fully design and characterize a high-rate, high-precision (and, of course, low cost) detector with limited time and resources meant that existing solutions had to be investigated. HPS adopted the use of *slic*[3] as its detector response simulation package. *slic* is a mature, highly flexible detector response simulation program based on Geant4[4], initially developed for Linear Collider simulations. Unlike most Geant4 applications in high energy physics which are detector-specific and often with hard-coded geometries and experiment-specific software dependencies, *slic* is a standalone executable. Full detectors (geometry, sensitive detectors, magnetic fields, ...) are defined at runtime via an input *lcdd*[5] xml file, an extension of GDML[6]. This meant that a single, well-debugged executable could be used to study multiple detector designs simply by providing different text file descriptions. The Geant4 hits in sensitive detectors, along with the Monte Carlo (MC) particle hierarchy, are written out using the LCIO[7] event data model (EDM) and persistency format. LCIO bindings exist for a number of computing languages, including python, C++ and Java, allowing HPS to employ whatever tools were most appropriate to the task at each stage (*e.g.* C++ for Geant4 simulations, Java for event reconstruction, Java/python/C++/root for physics analysis).

The overall design of the detector follows from the kinematics of A' production which typically results in a final state particle within a few degrees of the incoming beam. The hit occupancies of sensors near the beam plane are high, so high-rate detectors, a fast trigger, and excellent time tagging are required to minimize their impact and detailed simulations of backgrounds are crucial to the success of the experiment. The final design placed the silicon microstrip sensors of the tracking detector a mere 0.5mm from the beam. Simulation of the readout and the event reconstruction itself are performed with the Java-based software package *hps-java*. The simulation of the silicon microstrip tracking detector readout includes full charge deposition, drift and diffusion in the silicon wafers, followed by a detailed simulation of the readout chip and associated electronics. Full accounting of the occupancies in the tracker and calorimeter and the calorimeter-based trigger was performed by overlaying

simulated beam backgrounds on to MC simulated physics "signal" events. The resulting HPS detector is shown schematically in Figure 2.

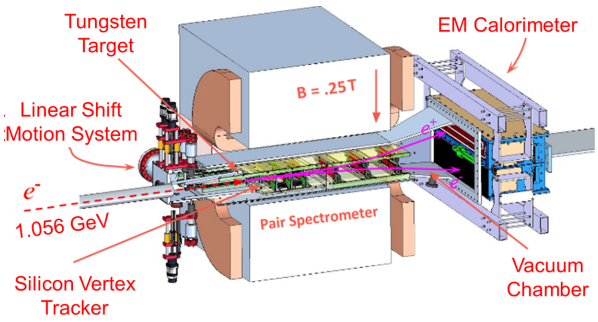

Figure 2. Schematic of the Heavy Photon Search Detector used during the engineering runs.

3 Event Reconstruction

The event reconstruction software is written in Java. The *hps-java* package builds upon the *lcsim.org*[8] reconstruction package originally developed for Linear Collider detector development studies and physics analyses. The same xml-based detector description used in the simulation is used in the reconstruction. *lcsim* provided complete access to the detector geometry, including navigation through arbitrary magnetic fields, access to run-dependent conditions databases, an event loop, and a suite of reconstruction packages for track-finding and fitting, calorimeter clustering, track-cluster association and vertex finding and fitting. It is also tightly integrated with the LCIO EDM and the *aida*[12] histogramming package. HPS is currently using track-finding and initial fitting inherited from the ILC *lcsim.org* packages. The final refit of track parameters is done with the General Broken Lines (GBL)[9] algorithm. The primary motivation for the use of GBL was its interface to the tracking detector alignment framework *millepede*[10]. Calling the C++ GBL library via the Java Native Interface (JNI) was cumbersome and slow, so the necessary features of the GBL package were ported to Java. Tests using both approaches were used to confirm identical results and using the native Java version led to a roughly 20% speedup in the track fitting. It also preserved the "write once, run anywhere" nature of the reconstruction software. Development work is currently underway to provide simultaneous pattern recognition and track fitting using a Kalman Filter approach for use in the 2019 physics data sample.The end result of the reconstruction is collections of tracks, calorimeter clusters and reconstructed particles (electrons, positrons, photons, and vertexed pairs of electrons and positrons, or V0s). These form the basis for further physics analysis. The well-defined and documented LCIO EDM and bindings to the LCIO file format in multiple languages gives the physicists broad leeway in choosing their physics analysis software.

The Java Analysis Studio[11] is used for data inspection and interactive analysis of histograms and ntuples saved in the *aida* format, whereas *root*[13] is used for root-based analyses. Both formats can be output from the *hps-java* analysis drivers. Wired4[14] is used for the event display, as shown in Figure 3.

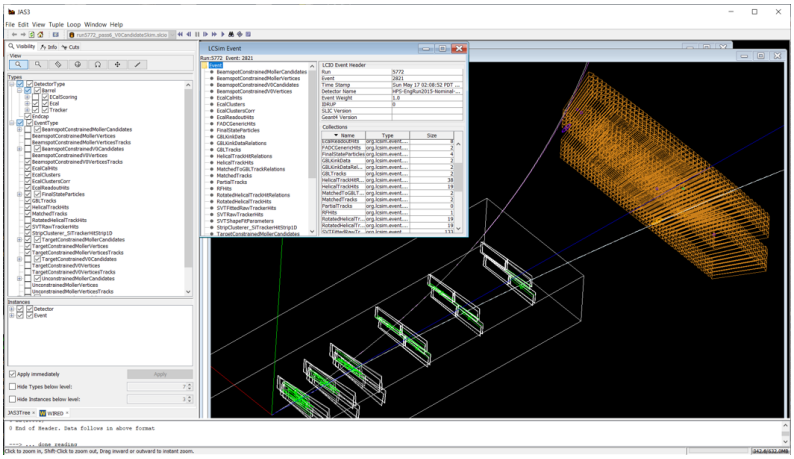

Figure 3. Both the interactive *lcsim* event browser and Wired4 event display are available as JAS3 plugins. Shown is a representative electron-positron pair with tracks reconstructed in the silicon microstrip detector matched to clusters in the electromagnetic calorimeter.

4 Code Management

git[15] is used as our code repository as well as our issue tracker and release manager. Projects for integrated development environments such as *netbeans*[16] and *eclipse*[17] are included in the Java software repository and these projects are configured to enforce style rules during editing. The Java-based simulation, reconstruction and analysis software is built using *maven*[18]. It is a constant cycle of compile, test and deploy, usually with several simultaneous, disparate *git* issues coexisting. To ensure correct behavior, unit tests with assertions are automatically run for individual classes and components. Integration tests run the full reconstruction over selected event samples. Analyses are automatically run over the resulting output and histograms are programmatically compared against reference histograms. Any differences in expected behavior are flagged as part of the release procedure. Once an issue has been resolved, two separate reviewers must approve the code before it is merged with the *git* master. Tagged releases are made when milestones have been achieved. Executable jar files are then deployed to a *nexus* repository. Being Java, the code simply runs on any platform with the Java Virtual Machine (JVM) installed, making distributed computing on multiple platforms straightforward. Production reconstruction of data or MC samples is performed with tagged jar files, which establishes a provenance for that data set.

5 Documentation and Communication

The *confluence*[19] wiki platform serves as our public portal and is used for most general-purpose documentation and communication within HPS. *slack*[20] is used for rapid interpersonal communication and quick turnaround for issue resolution. *indico*[21] is used primarily for agenda-setting and presentation-hosting for meetings. Technical notes, analysis notes and other more formal documentation reside in a series of *git* repositories. *Overleaf*[22] provides a very nice collaborative online LaTeX editor with versioned backups that integrates nicely with *git*. It is being used quite extensively within HPS (and was used to write these proceedings). In all of these choices we were driven to use solutions which were supported

by our host labs. Licenses and support for all of these tools were provided by either SLAC or JLab.

6 Summary

Small, heterogeneous collaborations cannot afford to invest significant manpower developing experiment-specific software solutions. Common, lightweight solutions need to be found which can be easily adopted and adapted and which do not impose a heavy burden on the individual collaboration members. The Heavy Photon Search experiment has successfully designed and built its detector and taken data using these software tools. The data from the 2015 test and 2016 engineering runs was fully reconstructed at JLab and analyzed by physicists at institutions across the US and in Europe. The performance of the detector during the 2015 test run along with comparisons to the Monte Carlo predictions was published in NIM[23] and the first physics results from 2016 were published in Physical Review D[24]. We are currently in the process of reconstructing and analyzing the data from our 2019 physics run.

Many of the benefits of developing collaborative, "generic" software (e.g. LCIO, *slic*, *lcsim.org*) have been realized in that software developed for Linear Collider detector simulations and physics analyses has been successfully adopted and used by a much smaller fixed-target experiment. Java-based reconstruction is working well for HPS (build once, run anywhere really works) and has eliminated the need to worry about operating systems, different compilers, different versions of runtime environments, *etc.*, all of which are all too common when dealing with distributed C++ development. Standard software development tools (*git*, *maven*, *nexus*) and procedures (such as automatic builds, unit and integration tests) make code development and deployment much easier and less error-prone. Modern documentation and communication software (*slack*, *confluence*, etc.) integrate well with each other and into the small experiment environment. Using existing, proven and available collaborative tools has enabled HPS to concentrate on the physics, not computer science.

References

[1] B. Holdom, Phys. Lett. **B166**, 196 (1986)

[2] James D. Bjorken, Rouven Essig, Philip Schuster, and Natalia Toro, Phys. Rev. D **80**, 075018 (2009)

[3] https://github.com/slaclab/slic

[4] Agostinelli *et al.*,Nuclear Inst. and Methods in Physics Research, A, **506**, Issue 3, 250-303 (2003)

[5] Norman Graf and Jeremy McCormick, Nuclear Inst. and Methods in Physics Research, A, **789**, 86-94 (2015)

[6] R. Chytracek, J. McCormick, W. Pokorski, G. Santin, IEEE Trans. Nucl. Sci., **53**, Issue: 5, Part 2, 2892-2896

[7] Computing in High Energy Physics and Nuclear Physics 2004, Interlaken, Switzerland, 27 Sep - 1 Oct 2004, pp.471 (CERN-2005-002) 10.5170/CERN-2005-002.471

[8] http://www.lcsim.org/

[9] Claus Kleinwort, Nuclear Inst. and Methods in Physics Research, A, **673**, 107-110 (2012)

[10] V. Blobel, Nuclear Inst. and Methods in Physics Research, A, **566**, 5-13 (2006)

[11] http://jas.freehep.org/jas3/

[12] http://aida.freehep.org/

[13] https://root.cern.ch/

[14] http://wired.freehep.org/
[15] https://git-scm.com/
[16] https://netbeans.org/
[17] https://www.eclipse.org/
[18] https://maven.apache.org/
[19] https://www.atlassian.com/software/confluence
[20] https://slack.com/
[21] https://getindico.io/
[22] https://www.overleaf.com/
[23] http://dx.doi.org/10.1016/j.nima.2014.12.017
[24] https://doi.org/10.1103/PhysRevD.98.091101