DgiStreamer: The Less You Code The More You Create

Mattia Angelini
DeepCamera
CYENS Centre of Excellence
Nicosia, Cyprus
m.angelini@cyens.org.cy

Simoni Panayi
DeepCamera
CYENS Centre of Excellence
Nicosia, Cyprus
s.panayi@cyens.org.cy

Alessandro Artusi
DeepCamera
CYENS Centre of Excellence
Nicosia, Cyprus
a.artusi@cyens.org.cy

***Abstract*—Building and deploying video pipelines is a task currently relegated to low level integration of specialized libraries, requiring thus a deep understanding of component functionalities and communication protocols. As a result, field experts, when undertaking this task, they often find the process laborious, filled with continuous coding checks and lengthy testing times. Introducing graphical visualization into the task can massively benefit imaging experts to view pipeline construction through a simpler lens, offering them a greater insight into the structure of their pipeline as well as better control over their model's workings. With this idea in mind, we provide an innovative solution through a GUI pipeline fabricator, called DgiStreamer, which allows easy construction and deployment of imaging pipelines via an effective and user-friendly interface, without writing a single line of code.**

***Index Terms*—Image Processing, Computer Vision, Imaging/video pipeline, Graphical User Interface.**

I. Introduction

Nowadays, there is a tremendous amount of visual data generated in the wild (around 3.2 billion images and 720K hours of video are shared online every day), courtesy of the billions of cameras and sensors generously available to the public [1]. Combined with the extraordinary strides that deep learning technologies have made over the past few years [2], the need for fully integrated tools that allow the construction of general pipelines together with state-of-the-art vision algorithm support is now greater than ever. Despite this ever-growing need, the task of building image processing and even audio pipelines is still presenting substantial overhead in implementation and testing to experts who currently depend on code API such as GStreamer [3], [4]. Firstly, working with such APIs is contingent on writing a considerable amount of code which renders the entire process rather laborious, which only becomes more unwieldy the more complex the desired pipeline gets.

One effective way to alleviate the image practitioner of the growing intractability of large pipelines is to introduce a visualization element. Since a pipeline can be represented as graphs then it can be naturally displayed graphically to the user to enhance their understanding [5]. An effort has already been made to achieve this but these tools are still few and limited in their scope.

To overcome this existing gap in the research community and industry, we propose our pipeline tool, DgiStreamer [6]. Through DgiStreamer, we provide a simple and effective solution for developing video pipelines, via Flow-based programming, allowing users to easily create their advanced pipelines while at the same time benefiting from all the already available functionalities offered by existing SDKs such as GStreamer [3] and Nvidia's Deepstream [7]. Through this tool, the image processing and computer vision practitioners can easily manage and overlook any future modifications to their imaging pipelines, much faster than any traditional approach. This enormous flexibility has the potential to transform the way traditional imaging practitioners and developers work today and DgiStreamer could be the pivotal element of a future ecosystem that will offer them a multitude of seamlessly integrated functionalities. In particular, the proposed tool will bring the following unique advantages and innovations to the state-of-the-art:

- **System agnostic**: The tool can be used in any machine, offering the choice to export the pipeline configuration or deploy it in a target system. As such, the application can run in a wide variety of Operating Systems, as opposed to the Deepstream Composer tool [8] developed from Nvidia which is only available for Linux and Windows.
- **Easy to Use**: A simple and intuitive interface, which makes it easy to create complex processing pipelines and quickly deploy them remotely to any any(device)-every(location)where, through SSH connectivity capabilities. This also includes both terminal functionality and a drag-and-drop functionality to easily interact with the underlying SDKs without having to write a single line of code.
- **Backward-compatibility**: The tool is agnostic to the type/version of the underlying SDK, automatically providing all the information related to the supported building blocks of the underlying SDK version. This will also allow maintaining backward compatibility with previous versions of the same underlying SDK.
- **Video Debugging System**: A video feedback system, useful for remote connections, is available to provide

[0]DgiStreamer© and DgiRemote© copyrights 2021

an immediate insight into whether the deployed pipeline exhibits the expected performances.
- **High Flexibility**: This is an intrinsic property of the product, leaving the users to concentrate on the high-level activities, e.g., no need to write any line of code and/or remember the type and parameters of the installed underlying blocks, and consequently increasing their productivity.

II. Related Work

As of today, tools exist that provide building blocks of various functionalities to develop a generic imaging/vision and audio pipelines. An example is GStreamer, a well-known tool in the community, which provides an exceptionally large selection of functionalities that cover numerous aspects of the classical steps of an imaging/vision as well as audio pipeline, funding use in a widespread range of applications [9], due its solid design. GStreamer currently provides a GUI based renderer, which is only able to render the already deployed pipelines. Additionally, the only tool dedicated towards pipeline building is a string-based property block builder which forces the user to remember beforehand low-level details such as which blocks are available within the supported SDK, how the blocks interconnect, and the entirety of each block's parameters. Moreover the creation of less linear fashioned pipelines is quite tricky and can easily become intractable.

Alternatively, Deepstream from Nvidia, which is built on the top of GStreamer and hence shares the same infrastructure, provides support for deep learning building blocks. However, GStreamer only provides extremely limited graphical interface capabilities to help practitioners and developers in building and deploying their pipelines. While Deepstream has announced a graphical tool to support construction [8], this is only dedicated to the deep learning blocks they provide, therefore introducing obvious constraints in the development of a generic imaging/vision pipeline. Most notably, it does not support all the GStreamer building blocks. Since the tool is based on the pure GStreamer run-time, it is able to support any block developed for it. This is in contrast to the Deepstream solution, which is tailored to the development of the pipeline using the Deepstream nodes, restricting its use towards AI powered applications.

III. DgiStreamer

The system was developed with the intention of providing practitioners with everything they need to construct and deploy their custom pipelines while being completely SDK- and platform-independent.

A. Architecture Design

We developed DgiStreamer on top of existing SDKs that provide an expanse of essential building blocks for constructing generic pipelines. In this work, we essentially show how our novel tool can pose as a holistic stand-alone pipeline constructor while at the same time providing the complete set of functionalities offered by the well-known SDKs such as GStreamer. To achieve this, our tool interfaces with the GStreamer run-time and deploys the generated pipeline converting it in a GStreamer configuration. By doing so, our design becomes completely separated from the GStreamer run-time and therefore can also exist in a different machine.

One of the most important features of our tool is our novel debugging system, DgiRemote, which provides information on the state of the pipeline and essential debugging details. The DgiRemote provides two main functionalities. First, it allows the system to be agnostic on which components of the underlying SDK are available and subsequently used in the process. This is achieved by collecting the system information and sending it to DgiStreamer (client) which in return provides the user with the whole graphical representation of the installed SDK blocks into the system. This enables backwards compatibility with older SDK versions and consequently gives the tool its agnostic nature. Additionally, having a detailed representation of the available blocks enabled us to create a bare-bone analyzer that allows the user to know which blocks can be connected together, therefore alleviating the burden of worrying about block availability and compatibility during pipeline construction. The second functionality of DgiRemote offers debugging capabilities, both remotely and locally depending on the configuration of the system, which makes the debugging process of the developed and deployed pipelines accessible and straightforward. The debugging functionality consists of a playback video server to stream back the output of the pipeline. Figure 1 shows how the video feedback is streamed in real-time back to the user for an immediate analysis of the pipeline output.

The user interface of our tool is created to depict the pipeline as a flowchart where each step can be described with a block (functionality) having a well-defined input and output. This can be described through a graph where nodes define the steps and the edges their order. Following this idea, the interface is built around the graph editing system, where the rest of the interface allows the user to customize the node setting and chose where to deploy and test the graphs.

B. Implementation details

The core of DgiStreamer was developed on top of the Electron [10] run-time, a versatile tool to develop cross-platform desktop applications with one code-base and web technologies. Together with the flexibility of Electron, we developed DgiStreamer as a Web Application, which provided access to the enormous breadth of tool and web development libraries and massively reduced the hands-on work load. The DgiRemote component was built and cross-compiled to work with different architectures and operating systems while simultaneously maintaining a fine-grained control over the final executable. This is especially important since IoT devices (like cameras or embedded systems) are limited in their computational power,

[0]This includes also DeepStream [7], which is an extension of the GStreamer SDK, providing deep-learning building blocks.

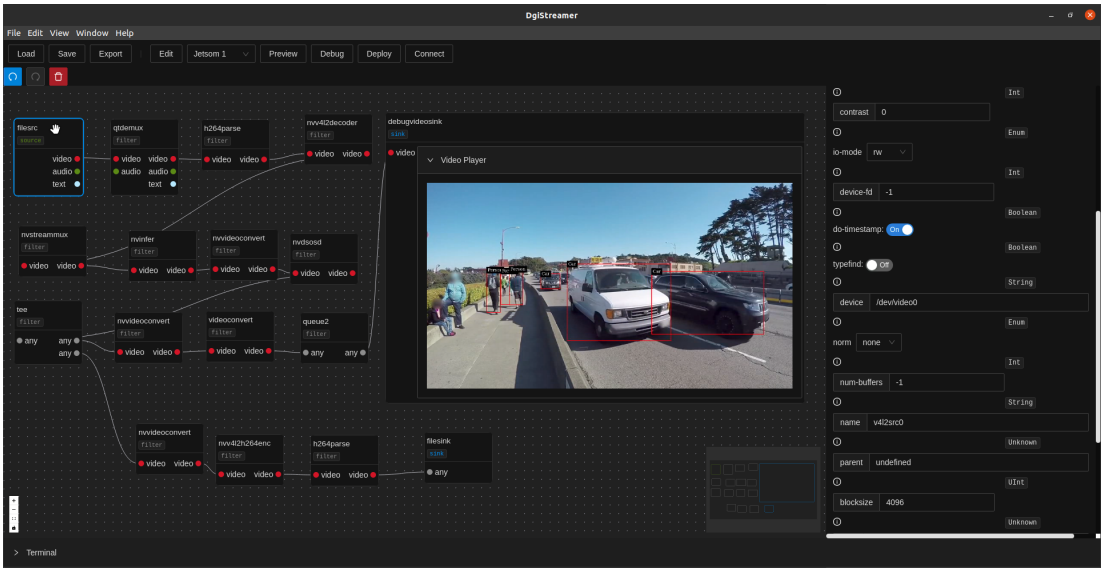

Fig. 1. DgiStreamer [6] - Example of constructed pipeline with video output feedback from DgiRemote component.

storage space and processing power therefore the ability to optimize the application for each individual device is a vastly beneficial feature of the DgiRemote component. As a result, our application provides the flexibility of using either the GStreamer CLI interface to communicate with the GStreamer run-time via Electron, or accessing it using our DgiRemote layer, allowing the user to set up the system however they see fit in their case. It should be noted that the DgiRemote provides extensive debugging and system information which helps avoid potential compatibility issues, making it a much more compelling feature for the user to exploit.

C. User Interface

The DgiStreamer interface has four identifiable core features:

1) Graph editor: Using the graph editor, the user can create the graph that connects the desired blocks with one another. In order to simplify the user experience, we decided to implement a colour coding system to represent the data types supported by each node. This simple design decision is in fact a much needed feature since GStreamer provides a multitude of video formats and compression standards that can lead to individual nodes supporting more than 20 different combinations of those. Therefore, the colour coding solution allows the user to keep track of the node-specific supported types. This of course provides the user with a more general idea of the data types supported by the nodes, which acts as an extra safety check on top of the more rigorous type checking system mentioned in Section III-A which prohibits linking of non-compatible nodes and identifies the block ports to which the selected nodes are allowed to be connected. In this case, despite the fact that both blocks glupload and avenc can work with data type video, the connection from the glupload to the avenc block is forbidden due to the fact that the two blocks work on different types of memory RAM, i.e. glupload on GPU and avenc on CPU RAM.

2) Node inspector: The node inspector where displays all information on the selected block. This information can range from meta-information regarding the authors and license of the blocks to the detailed information of the supported data formats of the node's input and output. Additional to the node information, we also provide the possibility to tweak the node parameters to provide the pipeline certain settings or configuration files which some nodes may require.

3) Terminal: The DgiStreamer interface also provides the user with a terminal in order to let the developers examine the output of GStreamer when the pipeline is deployed via SSH technology. The decision to provide an integrated Terminal interface was taken with the more devoted Terminal users in mind who have been accustomed to working with it when running GStreamer projects.

4) Tool bar: We have also added a useful tool bar on the top of the interface from which the user can choose the pipeline's deploy target via the dedicated drop down menu. In this way the user can chose to deploy the pipeline locally or on a remote machine in which GStreamer is installed.

IV. Use Cases

Given the abstracting nature of DgiStreamer, users can focus on the pipeline functionality without worrying about manual coding and API details, therefore we believe that the tool will thrive in the area of prototyping. With our tool at hand, the only work the user has to do is connect the desired nodes with each other and set up the appropriate parameters. During prototyping, experts can also have the comfort of not having to be on site to test and debug their work. For example, the user can use the full functionality of the tool remotely since they are given the possibility to sync the available nodes with the remote machine. While connected remotely, one is provided

with a detailed representation of the node library on the target machine, allowing them to build their pipeline using only the target machine-compatible nodes therefore avoiding any possible compatibility and versioning issues.

The deployment system provides also a tremendous flexibility in test-running one's prototype application. This is because DgiStreamer allows deployment of the same pipeline on different systems and offers the possibility of receiving visual feedback from the pipeline's video. The ability to do so allows the user to evaluate their work on many devices and analyse the performances.

An additional benefit of DgiStreamer is that it includes any plugin made available from GStreamer which also include all Nvidia Deepstream nodes. This enables the users to create seamless AI-inspired applications built on top of streaming pipelines that utilize video-, audio-, and image-based state-of-the-art deep learning algorithms.

To showcase our product's versatility and demonstrate its usefulness, we applied DgiStreamer as a solution to different scenarios that resemble possible real-life projects. The following use-cases serve as excellent examples of exploiting the full scope of DgiStreamer's features:

A. *Jeston© Powered Robot*

In this experiment, we developed an object detection system, powered by the Deepstream SDK, to allow the Unitree A1 dog robot to "see" and "recognise" its environment. Because the system itself supports the Deepstream SDK and GStreamer as extensions, the tool could easily interface to the device via IP, making the deployment and testing of the developed pipeline streamlined and effortless. Using the debugging functionality, we were able to stream back the visual feedback of the robot and see through its camera-eyes.

B. *Raspberry Pi© camera capture*

We were able to create a centralized processing server that takes as input the output of different cameras and applies any desired image processing methods to it. This was done by developing a multi-level pipeline, where we captured the video input from a Raspberry Pi and sent it to a remote server for further processing using the Remote Desktop Protocol (RDP) streaming technology.

V. Conclusion

Our tool is able to bridge the gap between pipeline construction and it's complete graphical representation, making its development and deployment a few clicks away. With DgiStreamer, the image processing and computer vision expert can easily develop and deploy their advanced pipeline in the machine of their choosing using the familiar user-favourite SDKs, GStreamer and Deepstream, without worrying about any compatibility issues. We also provide the comfort of a debugging functionality that can enhance the workflow of the project.

As it stands, DgiStreamer already has a lot of benefits to offer practitioners. However, we constantly seek to improve our tool in an effort to reach an ecosystem of seamlessly integrated functionalities that abstract as much as possible the low-level details and encourage practitioners to cultivate their high-level ideas. Some future improvements already in the works include:

- **Advanced Debugging**: A refined integration of custom GUI elements that can reveal to the user a more descriptive debugging information on the run-time.
- **Node management**: A dedicated editor for nodes with advanced configurations, e.g. the Deepstream nodes, which require an extended configuration for the Deep Learning inference block.
- **Decentralized**: An abstracting system for developing and deploying decentralized pipelines on different machines. This will be achieved with the addition of a cloud functionality that can handle groups of interacting IoT devices, enabling complex multi-machine projects.
- **Script2Graphs and Graphs import**: This is a backward compatibility feature allowing the conversion of existing complex pipeline scripts into their graphical representation readable by the tool, as well as importing of existing graphical representation into it.
- **Multiplatform**: The proposed tool will be multiplatform, i.e., PC, embedded, tablet and mobile-phone.

VI. Acknowledgements

This work has been partly supported by the project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 739578 (RISE – Call: H2020-WIDESPREAD-01-2016-2017-TeamingPhase2) and the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

References

[1] T. Thomson, D. Angus, P. Dootson, E. Hurcombe, and A. Smith, "Visual mis/disinformation in journalism and public communications: current verification practices, challenges, and future opportunities," *Journalism Practice*, pp. 1–25, 2020.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] "Gstreamer open source multimedia framework," Feb 2022. [Online]. Available: https://gstreamer.freedesktop.org/

[4] J. Newmarch, "Gstreamer," in *Linux Sound Programming*. Springer, 2017, pp. 211–221.

[5] J. Klerkx, K. Verbert, and E. Duval, "Enhancing learning with visualization techniques," in *Handbook of research on educational communications and technology*. Springer, 2014, pp. 791–807.

[6] "Dgistreamer v.0.1," May 2021. [Online]. Available: https://www.dgistreamer.com

[7] Nvidia, "Deepstream," Nov 2021. [Online]. Available: https://developer.nvidia.com/deepstream-sdk

[8] ——, "Deepstream composer," 2022. [Online]. Available: https://developer.nvidia.com/deepstream-getting-started

[9] K. Cannon, S. Caudill, C. Chan, B. Cousins, J. D. Creighton, B. Ewing, H. Fong, P. Godwin, C. Hanna, S. Hooper *et al.*, "Gstlal: A software framework for gravitational wave discovery," *SoftwareX*, vol. 14, p. 100680, 2021.

[10] O. Foundation, "Electron," 2022. [Online]. Available: https://www.electronjs.org/