NNFlowVector

Table of Contents

Preamble

1.1 Disclaimer

This document and the software is a work in progress, Pixelmania reserves the right to update and change the procedures and processes described in this documentation at any time. We will try our best to inform you when these changes happen with the hopes of minimising any impact on production.

1.2 Copyright

In addition to those names set forth on this page, the names of other actual companies and products mentioned in this User Guide (including and not limited to, those set forth below) may be the trademarks or service marks, or registered trademarks or service marks, of their respective owners in the United States and/or other countries, states and/or provinces. No association with any company or product is intended or inferred by the mention of its name in this User Guide.

Nuke is a trademark of The Foundry Visionmongers Ltd.
Linux is a registered trademark of Linus Torvalds.
CentOS is a trademark of Red Hat, Inc.
Rocky Linux is a trademark of Rocky Enterprise Software Foundation (RESF).
"Microsoft (R) Windows (R)" is a registered trademark of Microsoft Corporation in the United States and other countries.
Reprise License Manager is a trademark of Reprise Software, Inc.
Vimeo is a trademark of Vimeo, Inc.
Nukepedia is a trademark of OHUfx Ltd.

1.3 Acknowledgements and citations

We like to thank the authors of the following research work and publications:

```
@misc{teed2020raft,
    title={RAFT: Recurrent All-Pairs Field Transforms for Optical Flow},
    author={Zachary Teed and Jia Deng},
    year={2020},
    eprint={2003.12039},
    archivePrefix={arXiv},
    primaryClass={cs.CV}
}

@inproceedings{zhang2022flow,
  title={Flow-Guided Transformer for Video Inpainting},
  author={Zhang, Kaidong and Fu, Jingjing and Liu, Dong},
  booktitle={European Conference on Computer Vision},
  pages={74--90},
  year={2022},
  organization={Springer}
}

@article{huang2022flowformer,
  title={{FlowFormer}: A Transformer Architecture for Optical Flow},
  author={Huang, Zhaoyang and Shi, Xiaoyu and Zhang, Chao and Wang, Qiang and Cheung,
  Ka Chun and Qin, Hongwei and Dai, Jifeng and Li, Hongsheng},
  journal={{ECCV}},
  year={2022}
}
```

We also want to personally thank Thomas E. Vaughan for his excellent “region-fill-dirichlet” repository, and his great willingness to share and improve his code base.

2. Installation

2.1 System Requirements
- 64 bit Linux Operating System (for example CentOS 7.x), or
- 64 bit Windows Operating System (for example Windows 10 Pro)
- Nuke 12.0 or higher with a valid license.
  Nuke Indie is supported from NNFlowVector v2.1.0 (to run the plugin in Nuke Indie, you have to use the latest version of Nuke13.x or Nuke14.x). Please note that Nuke15.0 is not supported yet.

2.2 Plugin Installation
NNFlowVector relies on the standard Nuke process for finding and using plug-ins. Hence you only need to unzip the files in the downloaded zip archive (that matches your OS, Nuke version and CUDA version) to a directory of your own choice.
If you are using Windows, you do need to set an additional environment variable, please see the info below.

You then append that full path to the **NUKE_PATH** environment variable. This will tell Nuke to load the supplied **init.py**, **menu.py** and **NNFlowVector.so** (**NNFlowVector.dll** on Windows) files at startup. To read more about the **NUKE_PATH** environment variable, visit the documentation page by Foundry:
https://learn.foundry.com/nuke/content/comp_environment/configuring_nuke/defining_nuke_plugin_path.html

Linux:
The simplest way to create this environment variable is to put a single line like this in your **.bashrc** file in your user's home folder:
export NUKE_PATH=/full/path/to/plugin/install/folder

Windows:
In Windows 10, you have to add the environment variable using the system preferences. The easiest way to get to the right place is using the search bar on the Start Menu. Search for "environment variables", and choose "**Edit the system environment variables**". Add the "**NUKE_PATH**" variable and set the value to the full path to the folder where the plugin is installed.

For a more detailed guide of how to do this, you can search on Google for "windows 10 set environment variable permanently", and you will get links to pages like:
https://www.architectryan.com/2018/08/31/how-to-change-environment-variables-on-windows-10/
https://www.opentechguides.com/how-to/article/windows-10/113/windows-10-set-path.html

An alternative way to using the **NUKE_PATH** environment variable to specify for Nuke where to look for plugins, is to use the python command:
nuke.pluginAddPath("/full/path/to/plugin/install/folder")
This can be added to your **init.py** file located in your user's "**.nuke**" folder. Read more at:
https://learn.foundry.com/nuke/developers/13.0/pythonreference/_autosummary/nuke.pluginAddPath.html

Important! In Windows, you also have to set the **PATH** environment variable (in addition to the NUKE_PATH talked about above) to the full path of the folder where the plugin is installed. This is needed for the plugin to be able to find the additional provided libraries that it is dependent upon (in Linux, this is handled automatically by the system). You shall not replace your current PATH environment variable set by the system, but make sure to append it to the list of paths already there. The different paths in the list are separated by semi-colons (;).

2.3 Environment Variables

PIXELMANIA_NNFLOWVECTOR_DONT_RENDER_USING_GUI_LICENSES
(This environment variable is new as of NNFlowVector v2.1.0.)
The default behaviour of the node in batch/render mode is to look for dedicated NNFlowVector render licenses. If it can't find a render license, it will instead look for a GUI license. If it finds a GUI license, it will use it to render the node's output. This behaviour is not preferred if you only got a few GUI licenses available, i.e. you don't want the render farm to steal the artists' GUI licenses. To enforce the node to not use GUI licenses, please set the environment variable PIXELMANIA_NNFLOWVECTOR_DONT_RENDER_USING_GUI_LICENSES to the value of "1".
Please note that if you already got a site license, you don't want to set this environment variable since you already got an unlimited number of floating GUI licenses, so it's fine for the node to use those for rendering as well as in GUI mode.

PIXELMANIA_SUPPRESS_KERNEL_WARNINGS
(This environment variable is new as of NNFlowVector v2.2.0.)
As of NNFlowVector v2.2.0, the node will print out a warning during creation of the node if it's being dependent on JIT-compiled processing kernels. Basically this is happening if you have a very modern GPU that is using a compute capability that is higher than the native compute capability support of the specific build of NNFlowVector that you are running. The warning is there to inform you (the first time you run the plugin) that the kernels will get JIT-compiled which will take about half an hour (and appear like the computer is just hanged, even if it's heavily processing). The node will also check the environment variable CUDA_CACHE_MAXSIZE to see that it's set with a good value that will make it possible for the JIT-compilation to succeed (please see below for more info about this).
With all the above in mind, the new environment variable PIXELMANIA_SUPPRESS_KERNEL_WARNINGS can turn these warning messages off if it's set to the value of "1".

CUDA_CACHE_MAXSIZE
If you are using one of the CUDA10.1 versions of the NNFlowVector plugin, and you are having a modern compute capability 8.0 or 8.6 GPU, i.e. the Ampere generation of cards (for example RTX3080Ti), you need to set the CUDA CACHE for JIT compiled kernels to a high value using the **CUDA_CACHE_MAXSIZE environment variable**. We recommend setting it to **4000000000** because that is close to the maximum limit of the setting (4Gb). You need to do this because you are using an older version of CUDA that was released before the 8.0/8.6 cards were released. You will then rely on JIT compiling the internal PTX code instead of using the natively compiled kernels for your GPU architecture. These

kernels will only need to be compiled once, and will be saved to the CUDA CACHE in your user's home directory. But for this operation to succeed the cache size needs to be set higher than the default value.
The kernel compilation usually takes around 20-30 minutes or so, and then you are set to use the plugin without needing to do this again. Be aware of this need though, since the process is silent and might be experienced as a strange hanging without any printouts, progress bars and such.

If you don't want to do all this, you should download and use one of the CUDA11.1/CUDA11.2/CUDA11.8 compiled versions of NNFlowVector instead. These will run directly and potentially even faster on your modern GPU.
The only downside then is if you are using Nuke13.x/Nuke14.x and Foundry's own machine learning toolset like the CopyCat node. Nuke 13.x is only shipped with support for CUDA10.1/cuDNN8.0.5, Nuke 14.0 is only shipped with support for CUDA11.1/cuDNN8.4.1 and Nuke 14.1 is only shipped with support for CUDA11.8/cuDNN8.4.1. If you are using NNFlowVector with another version of CUDA/cuDNN you can not run them in the same script without problems/crashing.

When it comes to the newly released Nuke 14.1, we are only shipping NNFlowVector compiled against the exact same CUDA version and cuDNN version as Nuke is using internally. This is because Foundry have updated to a CUDA version that natively supports the RTX40xx series of NVIDIA GPUs. NNFlowVector for Nuke 14.1 is hence using CUDA 11.8 and cuDNN 8.4.1 to fully be compatible with Nuke's own AIR tools.

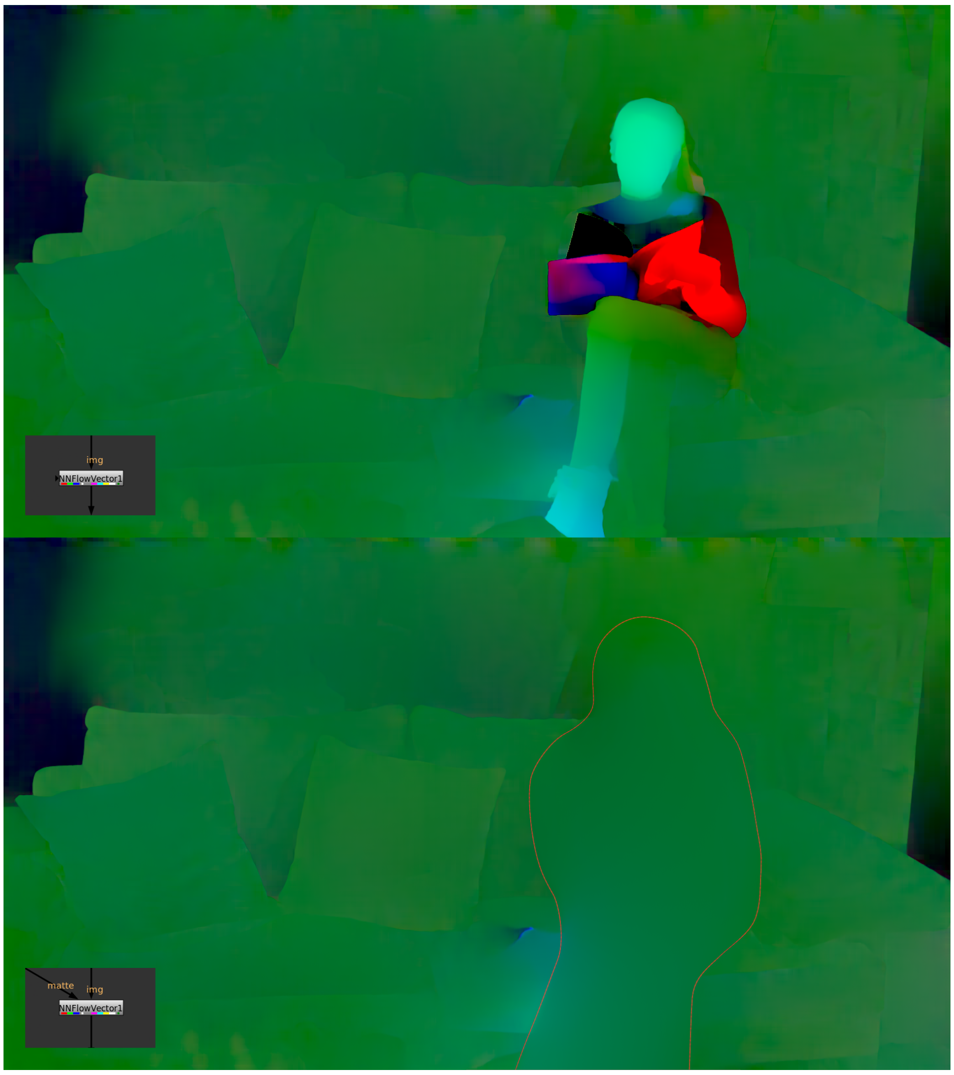
3. Usage and Knob Reference

3.1 Usage of the Node

The NNFlowVector node can produce normal motion vectors that are compatible with how Nuke handles motion vectors, i.e. they become available in the "motion" layer and also as a subset in the "forward" and "backward" layers. These can be used in native Nuke nodes such as VectorBlur, among other third party tools from Nukepedia (www.nukepedia.com). The motion vector output is available, without limits, in the free version of NNFlowVector.

Worth mentioning is that you do want to set the "interleave" option in the Write node to "channels, layers and views" when pre-comping out motion vectors (this is Nuke's default settings). This way the "forward" and "backward" layers are combined automatically by Nuke to be represented in the "motion" layer as well. If you write out the files with "interleave" set to for example "channels" only, then the "forward" and "backwards" layers become written out in separate parts in the resulting multi-part EXR sequence. This results in Nuke failing to recombine them correctly to the "motion" layer when read in again. You can easily see if this has happened by checking if the "motion_extra" layer has been created instead of the standard "motion" layer.

SmartVector, and the tools that come bundled with it in NukeX, is truly a brilliant invention that opens of for a lot of powerful compositing techniques. The VectorDistort node is the most obvious one which makes it possible to track on images, or image sequences, onto moving and deforming objects in plates. This is made possible by the more complex set of motion vectors called smart vectors. While NukeX is able to produce these smart vectors, they are quite often pretty rough which makes them not hold up for that long sequences. Basically the material you are tracking onto the plates does deteriorate over time. This is



where NNFlowVector comes in by being able to produce cleaner and more stable motion vectors in the smart vector format. Hence the output of NNFlowVector is usable directly in the VectorDistort node. This is also true for other smart vector compatible nodes like VectorCornerPin and GridWarpTracker.
You have to use the licensed/paid version of NNFlowVector to be able to produce smart vectors. If you want to try this feature out first, there are free trial licenses available upon request on our website www.pixelmania.se.

New in v2.0: Matte input
The new matte input is a powerful feature where you can select an area to ignore when the plugin is generating the motion vectors. It works by providing the plugin with an alpha channel using the “matte” input of the node. It will then inpaint and enhance the specified area when the vectors are being produced, resulting in vectors representing the motion of what should have been there if the object in question wasn’t present. This makes is possible, for example, to track image patches onto a wall behind a character that crosses in front of it. Please see the image above for an example of using the matte input (the red lines are just the roto splines, they are not part of the produced vector output).

3.2 Knob Reference

The knobs “max size”, “padding” and “overlap” are mostly related to being able to generate motion vectors for pretty large resolution sequences using limited VRAM on the graphics card. The neural network requires pretty large amounts of memory even for small resolution input sequences. To be able to generate vectors for a full HD sequence or larger, the images most likely need to be split up into several passes/patches to fit into memory. This is all done and handled transparently “under the hood” so the artist can focus on more important things. You might need to tweak the settings though depending on the use case and available hardware on your workstation.

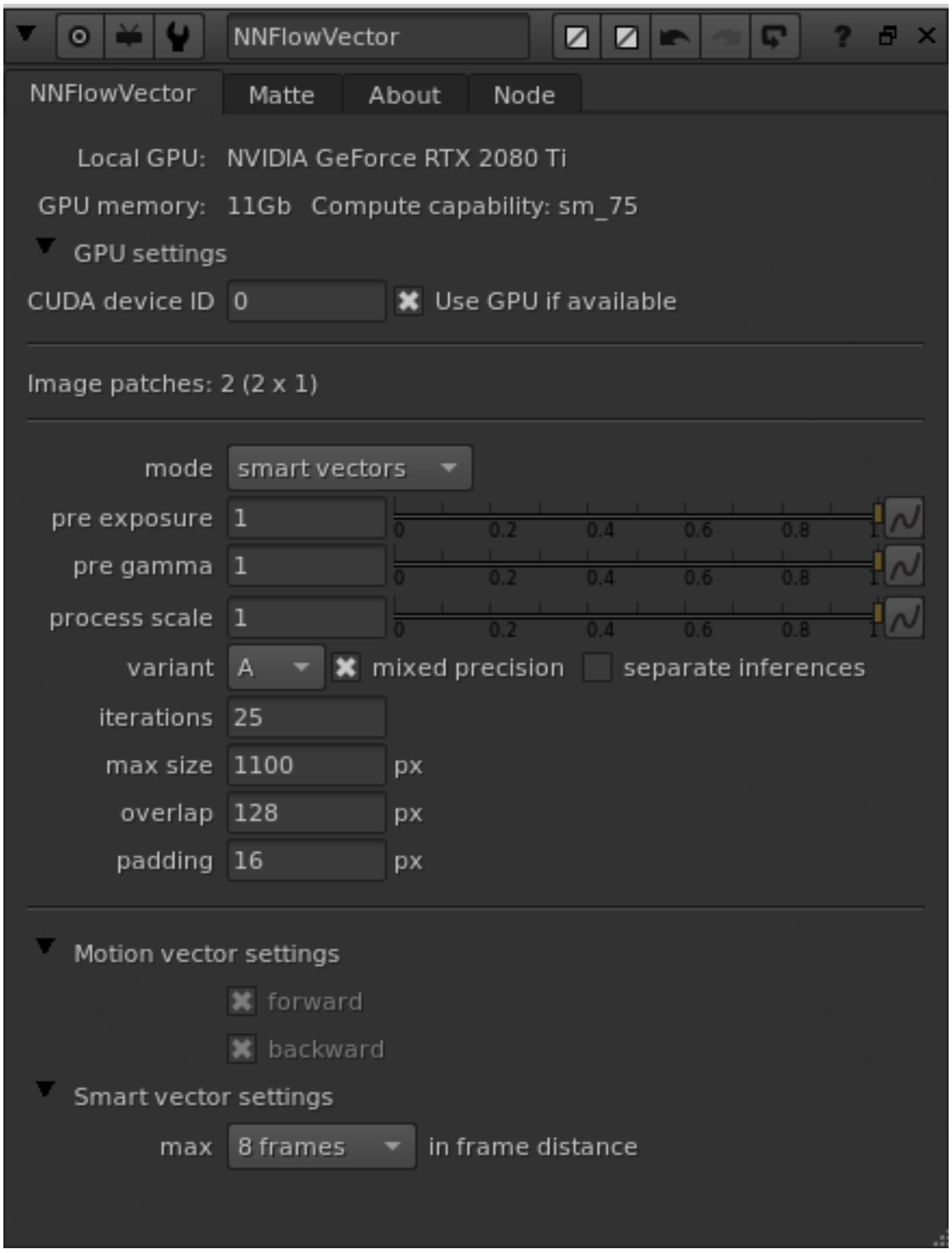
mode
You can switch between generating normal “motion vectors”, and the more Nuke specific “smart vectors”, depending on your need and what nodes you are going to use the motion vectors with.

pre exposure
This is a normal exposure grade correction that is applied before the neural network processing. It is here for artist convenience since it’s worth playing a bit with the exposure balance to optimise your output.

pre gamma
This is a normal gamma grade correction that is applied before the neural network processing. It is here for artist convenience since it’s worth playing a bit with the gamma balance to optimise your output.

process scale
This is a new knob since v1.5.0. Defines in what resolution to calculate the vectors in. The default value of 1.0 means it’s going to calculate the vectors in the same resolution as the input images (basically this ignores doing a pre-scale). A value of 0.5 means half res, and a value of 2.0 means double res. It’s good being able to alter the process resolution because of rendering speed, memory use and produced details. For example it’s quite often enough to calculate the vectors of UHD material in full HD instead (as in setting the process scale to 0.5). The vectors are always automatically reformatted back to the input resolution and scaled accordingly, so they are directly usable together with the input material. Please have a play with your particular material to find the most optimal process scale setting.

variant
This is a new knob since v1.5.0. The variant, A to H, sets what neural network training variant to use. **As of NNFlowVector v2.1.0, you also have a new transformer based model available using the “AA” variant**. This new model does often produce even better vectors for tricky image sequences compared to the normal A to H variants. It will however use more VRAM on your GPU to generate the vectors. To be able to fit the model and processing you might need to lower the “max size” knob value. You can also test enabling the new knob “separate inferences”, see below for more info.
We provide the NNFlowVector plugin with 9 different training variants made with slightly different optimizations and training settings. The different alternatives does produce pretty similar results, but a particular one could produce a favourable result with your particular material. If you got the processing power and time, you could try them all out to really get


NNFlowVector

NNFlowVector | Matte | About | Node

Local GPU: NVIDIA GeForce RTX 2080 Ti

GPU memory: 11Gb Compute capability: sm_75

GPU settings

CUDA device ID 0 [✓ Use GPU if available]

Image patches: 2 (2 x 1)

mode: smart vectors

pre exposure: 1

pre gamma: 1

process scale: 1

variant: A [✓ mixed precision] separate inferences

iterations: 25

max size: 1100 px

overlap: 128 px

padding: 16 px

Motion vector settings

[✓ forward]

[✓ backward]

Smart vector settings

max 8 frames in frame distance

the best possible result. If you are unsure and just want to quickly get a good result, please use the default variant of “A”, or try the new “AA” variant.

mixed precision
This is a new knob since v1.5.0. When this is “on" (which is default) the plugin will try to use mixed precision on the GPU for calculations. What this means is that for certain supported calculations, the GPU will process data using half floats (16 bit) instead of normal full floats (32 bit). This results in a bit less VRAM usage and faster processing, for the little cost of a slightly less accurate result. You usually won’t notice the difference in quality at all, but the resulting processing speed can be, for example, 15% faster. The actual speed difference depends on your exact GPU model.
If you are getting CUDA processing errors, please try to process the same material but turning off mixed precision.

separate inferences
This is a new knob since v2.1.0. When “separate inferences” is on, the calculation of the forward motion vectors and the backward motion vectors are run as separate inference/calculation passes through the neural network. This takes a little bit more time, but uses less GPU memory which (dependent on your hardware) might be a good thing. The default (having this checkbox off) is to calculate the forward and backward vectors in parallell using one inference run through the neural network (a touch faster, but uses more memory).

iterations
Defines how many refinement iterations the solve algorithm will do. This knob has been re-implemented as a free typed integer as of v.1.5.0 (before that it was a static drop down menu). The default has been increased to 25 (it was earlier set to 15). You can go really high on the iterations if you like, like 100 or more, but that doesn’t necessarily produce a better result. Please have a play with your particular material to find the most optimal iterations setting.
Please note that this knob doesn’t affect the new “AA” variant, hence it’s greyed out when that variant is used.

max size
Max size sets the maximum dimension, in one direction, that an image patch can have and is one of the most important knobs on the plugin. The default is 1100, which means that the max dimensions an input patch would be allowed to be is 1100×1100 pixels. From our experience that will use up to around 8Gb of VRAM on your graphics card (applies to the A-H variants. For the new “AA” variant, please try values in the realm of 700 instead). If you haven’t got that available and free the processing will error out with a CUDA memory error, and the node in Nuke will error in the DAG. To remedy this, and also to be able to input resolutions much higher than a single patch size, you can tweak the max size knob to adapt to your situation. You can lower it to adapt to having much less VRAM available. The plugin will split the input image into lots of smaller patches and stitch them together in the background. This will of course be slower, but it will make it possible to still run and produce much larger results. There is a text status knob above the “max size” knob (in between the two dividers), that will let you know how many image patches the plugin will run to create the final motion vector output.

Since motion vectors do describe local and global movements in plates, they are pretty sensitive to what image regions are part of the algorithm solve. What this mean is that the more it sees the better result it will be able to produce. Keeping the max size as high as you can, will produce better motion vectors. It’s worth to mention that this is much more sensitive in a plugin like this compared to for example NNSuperResolution.

overlap
Since the plugin will run multiple patches through the neural network, there will be lots of edges of image patches present. The edges doesn’t get as satisfying results as a bit further into an image from the edge. Because of this, all these multiple patches that get processed are done so with some spatial overlap. The overlap knob value sets the number of pixels of the patches’ overlap. The default value is 128, which is usually a good start, but sometimes higher values like 256 is needed to get a good result.

padding
The padding is very connected to the overlap above. While the overlap sets the total amount of pixels the patches overlap, the padding then reduces the actual cross fading area where the patches are blended to not use the very edge pixels at all. The padding is also specified in pixels, and the default value is 16. This way of blending all patches together has proven pretty successful in our own testing.

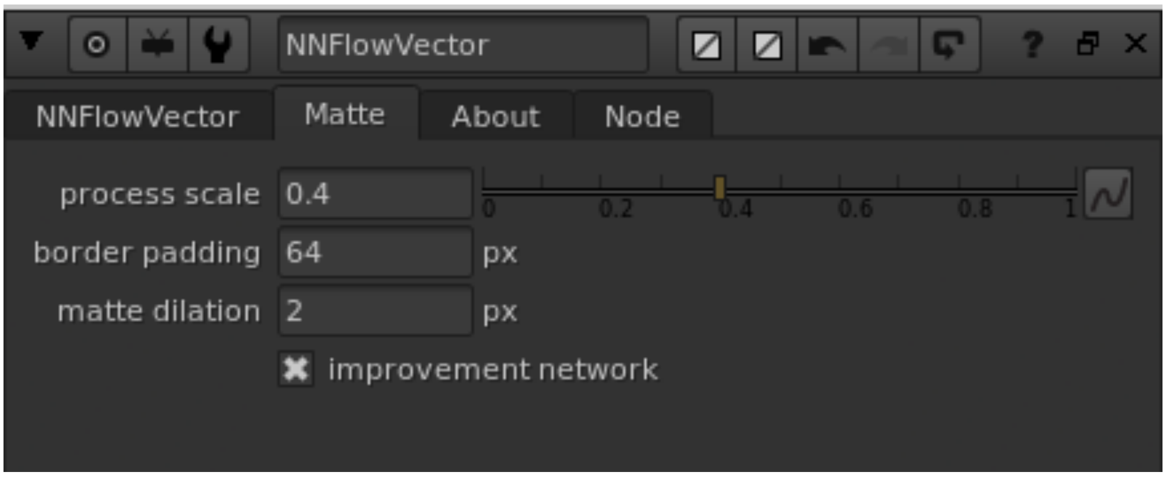
motion vector mode: forward and backward
Motion vectors describe the local movement of different parts of the plate from the current frame to the next frame (forward), and from the current frame to the frame before (backward). Different tools and algorithms have different needs for what to use in regards to motion vectors, some work with just forward vectors and some need both forward and backward vectors. You have the option of which ones you want to calculate, but we recommend to always calculate and save both.
Worth noting is that all the NNFlowVector Util nodes that got motion vectors as input (please see below in this document) do require both forward and backward vectors.

smart vector mode: frame distance
This knob has been re-implemented as a drop down menu as of v1.5.0 (it was earlier a bunch of checkboxes which made it much more confusing). The drop down menu specifies the highest value of “frame distance” that you can use in a VectorDistort node when using the rendered SmartVector compatible output of NNFlowVector. If you want to have all the possible options available in a VectorDistort node, you should set the frame distance to 64. This will require a lot more rendering time though. If you know that you will only use up to a frame distance of 2, for example, you can optimise the rendering of your smart vector compatible output a lot by only rendering the needed motion layers for that particular setting (and hence set the max frame distance in NNFlowVector to the same setting of 2).

CUDA device ID
This is a new knob since v1.5.0, and it’s only present if you’ve installed a GPU version of the plugin. This knob specifies what GPU to use in the system by its CUDA device ID. It’s only relevant if you got multiple GPUs installed in the system. The default value is 0, which is the default CUDA processing device, usually the fastest/most modern GPU installed. Please refer to the output of running the command “nvidia-smi” in a terminal for retrieving the info of the specific GPU device IDs you have assigned to your GPUs in your particular system.

Use GPU if available
There is another knob at the top of the plugin called “Use GPU if available”, and it’s “on” by default (recommended). This knob is only present if you’ve installed a GPU version of the plugin. This knob is not changing the motion vector output, but rather how the result is calculated. If it is “on” the algorithm will run on the GPU hardware of the workstation, and if it’s “off” the algorithm will run on the normal CPU of the workstation. If the plugin can’t detect a CUDA compatible GPU, this knob will automatically be disabled/greyed out. This knob is similar to the one you’ll find in Foundry’s own GPU accelerated plugins like for example the ZDefocus node.

We highly recommend always having this knob “on”, since the algorithm will run A LOT faster on the GPU than on the CPU. To give you an idea of the difference, we’ve seen calculation times of the same input image be around 2.0 seconds/frame using the GPU and about 34 seconds/frame on the CPU (a difference factor or 17x slower on the CPU). These numbers are just a simple example to show the vastly different processing times you will get using the GPU vs. the CPU. For speed references of processing, please download and test run the plugin on your own hardware/system.


NNFlowVector

NNFlowVector | Matte | About | Node

process scale 0.4
border padding 64 px
matte dilation 2 px
✖ improvement network

The “Matte” tab’s knobs:

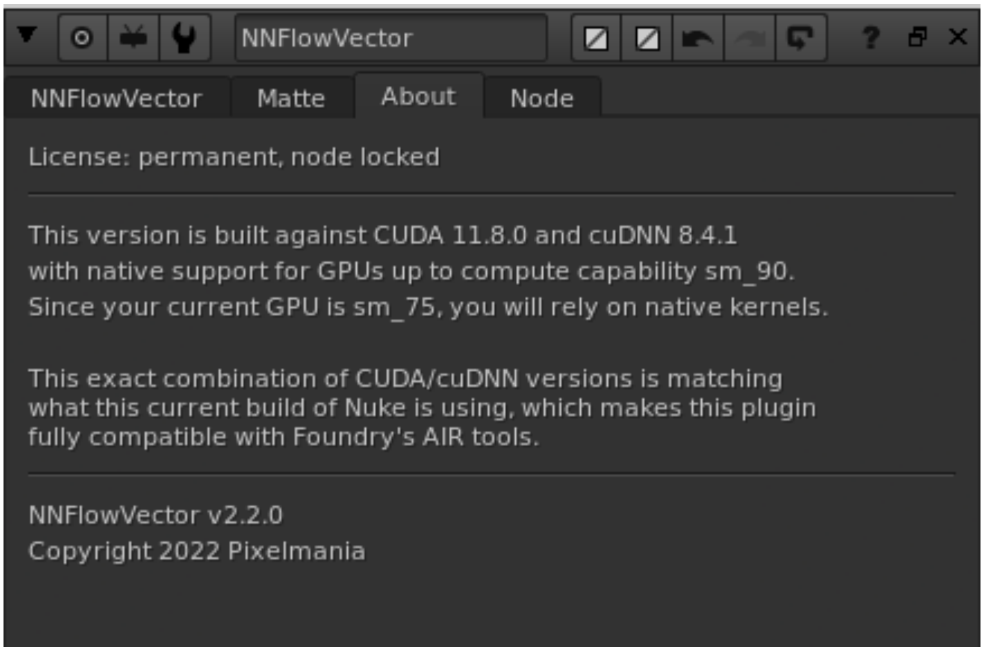
process scale
This knob specifies at what resolution the matte region should be calculated in, i.e. the calculation of the inpainting and enhancement of the “ignored” matte region specified by the alpha channel of the matte input. This process is rather compute and memory heavy, so it’s good to keep this at a rather low ratio to not run out of CUDA memory (GPU VRAM). The default value is 0.4.

border padding
The border padding setting is only relevant if your input matte is touching any of the edges of the image. If it is, then some extra inpainting steps needs to be performed and it’s here the “border padding” is relevant. If you are getting artefacts at the image edge in the matte region, such as black bleed-ins, then try to increase/change the border padding. The default value is 64 pixels.

matte dilation
The matte dilation setting is specifying how many pixels to dilate the matte before it’s used for the inpainting and improvement neural network. Because the original matte is used to actually keymix in the calculated vectors in the matte region, this dilation setting makes the end result not use the furthest out pixels produced by the algorithms. The setting basically makes is possible to ignore the outer edge of the produced matte region which is good because it can have some noisy artefacts in some cases. Keep this value as low as possible, even set it to 0, as long as you’re not experiencing the edge artefacts. If you get them, slowly increase this setting. The default value is 2 pixels.

improvement network
This setting should pretty much always be “on”. It turns the improvement neural network on/off. If it’s set to “off” you will instead only get the pre-processing done before the vectors

are fed to the neural network. This will be similar to what a classic “Inpainting” node in Nuke looks like, or a “Laplace region fill” if you’re more into math.



License: permanent, node locked

This version is built against CUDA 11.8.0 and cuDNN 8.4.1
with native support for GPUs up to compute capability sm_90.
Since your current GPU is sm_75, you will rely on native kernels.

This exact combination of CUDA/cuDNN versions is matching
what this current build of Nuke is using, which makes this plugin
fully compatible with Foundry's AIR tools.

NNFlowVector v2.2.0
Copyright 2022 Pixelmania

The “About” tab:
There is an additional tab called “About” in the knob panel for NNFlowVector. This tab prints out some additional info that can be very helpful for trouble shooting. The first section is printing out what type of license that is currently in use, and also how many more days it’s valid for.
The next section (if you are running a GPU based build) is printing out what CUDA and cuDNN this version is built against. It also lists what compute capability your current GPU is supporting and what max compute capability the current build of the plugin is natively supporting. If also writes out if your current configuration will use native or JIT-compiled kernels. In addition to that, it also writes out if the current plugin build is matching what versions of CUDA and cuDNN that the current Nuke build is using, and hence if it’s compatible with the AIR tools or not.
Finally it’s listing what version of the plugin that is running.
(the screenshot above is just a snapshot of how this info can look, captured on one of our development workstations)

4. Bundled utility tools

4.1 Overview

NNFlowVector comes bundled with a few utility tools, collectively called “NNFlowVector Utils”. These tools use the vector output of NNFlowVector to help you with certain compositing tasks. The tools are meant to encourage artists to use good quality motion vectors for more than adding motion blur to material, to show the vast possibility of things that are doable. The tools do not try and be a complete tool set for all the things that motion vectors/smart vectors can be used for, but rather as a nice additional collection of utils that could come in handy quite often.

4.2 Utilities

These are the tools that together make up “NNFlowVector Utils”:

MotionVector_to_Tracker
This tool can create a Tracker node with up to four tracked points from anywhere in the input material. It expects an image sequence as input that carry the “motion” layer. It helps if you also got the RGB layer, because then you can view the images when working with the tool. This is the standard output way from NNFlowVector, i.e. if you pre-comp out the result of that node when set to generate “motion vectors”, it will output both the RGB channels for the material as well as the generated “motion” layer.
You then find a good frame where the points are visible that you want to track. You enable as many points you need, and place them where necessary on the image. You then press the button “bake and create tracker”, which will calculate the tracks for the specified frame range, and finally create a new Tracker node in the DAG that is ready to use somewhere in your comp.

SmartVector_to_Tracker
This tool is very similar to the tool “MotionVector_to_Tracker” mentioned above, except that it instead expects an input that carry all the smart vector layers. The rest of the tool works exactly the same.

MotionVector_to_Roto
This tool can automatically animate Roto shapes and RotoPaint brush strokes to follow movement of objects in plates (using the motion vectors from a NNFlowVector node). You can either apply animation to all the individual vertices or as a global translation to the shapes/strokes. If you apply the animation to the individual vertices, you do get the shapes to deform to the local movement over time, and not just track the overall movement.

It works as follows:
You first create a Roto node and draw the roto shape you need, or using a RotoPaint node to paint what is needed using the normal brushes. You can key frame a couple of important shapes over time to help the solve if you want. Create a “MotionVector_to_Roto” node and connect it to a pre-comped output from NNFlowVector having motion vectors. When having both the node panels open of the “MotionVector_to_Roto” node and the “Roto”/“RotoPaint” node, do select the “Roto”/“RotoPaint” node in the DAG and also select the shapes/strokes you want the animation to applied to in the node. In the “MotionVector_to_Roto” parameters, use the solve buttons at the bottom to either solve a frame range between key frames at a time or the “ALL” button to solve the full frame range specified in the node.

MotionVector_FrameBlend
This tool can create a frame blend of neighbouring frames, either 3 or 5 frames (3 means 1 frame on each side of the current frame, and 5 means 2 frames on each side of the current frame). The powerful thing with it is that before blending the frames, they are individually distorted

using the motion vectors to match to the current frame, i.e. all frames are pre-warped to the current frame before used. You can choose between “average” and “median” as blending modes. One use case for this would be to smooth out flickering material (average), and another one would be to remove small and fast moving objects like snow flakes (median).

MotionVector_DistortFrame
This tool uses the motion vectors to distort a neighbouring frame to the current one. You can choose which frame to use with the “which” parameter. The “which” parameter got a valid range of -2 to 2. One use case for this node would be to distort a neighbouring frame to the current one when doing cleanup of small objects passing by such as wires.

SmartVector_to_MotionVector
This tool converts Nuke compatible smart vectors to normal motion vectors. The smart vectors are a more complex type of motion vectors used by several advanced NukeX nodes, such as VectorDistort. They do however also contain normal motion vectors which can be extracted, so you can use them with tools that expect a motion vector input (like for example many of the NNFlowVector Utils nodes mentioned above). If you already got pre-comped smart vectors from the NNFlowVector node, it’s much faster to convert them to motion vectors than rendering a new motion vector output from NNFlowVector.

5. Frequently Asked Questions

Is Nuke Indie supported?
Yes, Nuke Indie is supported from v2.1.0. To have the plugin working in Nuke Indie you have to use the latest version available from Foundry of Nuke 13.x or Nuke 14.x.

How are licenses consumed?
All of Pixelmania’s licenses are per host, i.e. you can have multiple jobs using NNFlowVector on the same host and only use a single license. If you query the license server it may report multiple handles for those jobs, but it’s still only using a single license token.
As of NNFlowVector v2.1.0 we do support dedicated render licenses as well. The default behaviour when rendering using Nuke’s batch mode (i.e. command line rendering), is to check out render licenses. If it can’t find a render license, it will instead try and check out a GUI license. This behaviour is preferred if you got a node locked license, or if you got a site license. It is not preferred if you have only a few GUI licenses and then a bunch of render licenses. To stop the node from using GUI licenses when rendering, you can set the environment variable
PIXELMANIA_NNFLOWVECTOR_DONT_RENDER_USING_GUI_LICENSES to the value of “1” (please see more about this in the section “2.3 Environment Variables”).

Do you offer educational discount?
As of now, we are not offering any default educational discount. We’ve chosen to sell our licenses for a low price instead to accommodate most people and companies to afford a

license anyway. But you are of course always welcome to contact us directly and state your case.

Am I able to transfer my NNFlowVector license to another machine?
Yes, please fill out the License Transfer Form available at this address:
https://www.pixelmania.se/license-transfer-form
The form should be filled in, signed, scanned and emailed to licenses@pixelmania.se, and we will send you a new license file as soon as we can.

Do you really need CUDA installed to be able to use the GPU?
You don't need to have the CUDA Toolkit installed to use our plugins and have them GPU accelerated. All of our plugin downloads come bundled with the needed CUDA and cuDNN libraries from NVIDIA. What you need to have them work correctly is a supported GPU (please see below) and a modern enough NVIDIA graphics driver version installed.

You can download the latest graphics drivers from NVIDIA's website here:
https://www.nvidia.com/Download/index.aspx

What NVIDIA graphic card architectures are supported?
The currently supported CUDA architectures are the following compute capabilities (see https://en.wikipedia.org/wiki/CUDA):

- 3.5 (Kepler)
- 5.0 (Maxwell)
- 5.2 (Maxwell)
- 6.0 (Pascal)
- 6.1 (Pascal)
- 7.0 (Volta)
- 7.5 (Turing)
- 8.6 (Ampere)*
- 8.9 (Ada Lovelace)**

* The Ampere architecture, i.e. the NVIDIA RTX30xx series of cards, works with all CUDA variants of NNFlowVector (**CUDA10.1, CUDA11.1, CUDA11.2, CUDA11.8)**. The CUDA11.1/CUDA11.2/CUDA11.8 versions got native support and will work directly. The CUDA10.1 versions will need to compile the CUDA kernels from the PTX code. Please see more info above in the environment variables section.

* The Ada Lovelace architecture, i.e. the NVIDIA RTX40xx series of cards, works with all CUDA variants of NNFlowVector (**CUDA10.1, CUDA11.1, CUDA11.2, CUDA11.8)**. The CUDA11.8 versions got native support and will work directly. The CUDA10.1/CUDA11.1/CUDA11.2 versions will need to compile the CUDA kernels from the PTX code. Please see more info above in the environment variables section.

Why is the plugin so large?
This is a result of including a lot of needed static libraries for neural network processing, and also for supporting lots of different graphic cards for acceleration. The plugin could have been much smaller if all these pieces of software used were required to be installed as dependencies on the system instead. That would kind of defeat the nice ecosystem of

having self contained plugins that work simply by themselves, hence the plugin need to be a rather large file.

I’m having problems running NNFlowVector in the same Nuke environment as KeenTools FaceTracker/FaceBuilder
We have noticed that trying to load both NNFlowVector and FaceTracker in the same Nuke environment makes the plugins clash with each other. The error you get is that NNFlowVector is not finding the GOMP_4.0 symbols in the libgomp.so shared library that is loaded. This is because both NNFlowVector and FaceTracker are using the libgomp.so shared library, but KeenTools have decided to ship their plugin with a version of this library. This specific library version they are shipping is a rather old version of the library. So what happens is that NNFlowVector is trying to use the version from FaceTracker but it’s expecting a newer version, and it’s failing. The fix we have found working well is simply to delete the libgomp.so file that is shipped with KeenTools. You probably already got a newer libgomp.so file installed on your system. If that is the case both NNFlowVector and FaceTracker will pick up the newer libgomp.so shared library from the system, and since this library is backwards compatible everything will work just fine. If you haven’t got libgomp.so installed on your system, you need to install it to run the plugins. How that is done is of course dependent on your operating system and such. On CentOS you can install it by simply running the command “yum install libgomp“.

I’m having problems running NNFlowVector in the same Nuke environment as Kognat’s Rotobot
Please make sure to match versions of both NNFlowVector and Rotobot to what exact CUDA version they have been built against. We have synced our efforts together with Kognat, and made sure that we both got compatible releases with each other, built against either **CUDA10.1/cuDNN8.0.5, CUDA11.2/cuDNN8.1.0, CUDA11.1/cuDNN8.4.1** or **CUDA11.8/cuDNN8.4.1**.

I’m having problems/crashing when using NNFlowVector together with Foundry’s AIR nodes in Nuke13.x
You are highly likely using the CUDA11.2/cuDNN8.1.0 build or the CUDA11.8/cuDNN8.4.1 build if this is happening. These builds of NNFlowVector are for modern GPUs like the RTX3080 or RTX4080 cards, and will work fast and nice except for when you need to process NNFlowVector output together with for example CopyCat nodes. If you need to support this, you have to use the CUDA10.1/cuDN8.0.5 build instead. Please see more info about this above in the environment variables section.

I’m having problems/crashing when using NNFlowVector together with Foundry’s AIR nodes in Nuke14.0
You are highly likely using the CUDA11.8/cuDNN8.4.1 build if this is happening. This build of NNFlowVector is for modern GPUs like the RTX4080 card, and will work fast and nice except for when you need to process NNFlowVector output together with for example CopyCat nodes. If you need to support this, you have to use the CUDA11.1/cuDN8.4.1 build instead. Please see more info about this above in the environment variables section.

6. Change Log

v2.2.1 - November 2023

- Corrected a bug that made the node crash when a GPU build was used to process on a machine without an installed GPU. Now it automatically falls back to using the CPU instead, as intended.
- Added support for Nuke15.0 (which also is natively built against CUDA 11.8 and cuDNN 8.4.1, on Rocky8). Released in January 2024.

v2.2.0 - November 2023

- More efficient releasing of used GPU memory while using the node in a live Nuke GUI session, i.e. you don't have to restart Nuke anymore to get some of the GPU memory back. Also doing a more thorough GPU memory cleanup when the node is fully deleted (usually when you run "File/Clear" in the Nuke GUI).
- Added native support for the NVIDIA RTX40xx series of GPUs, i.e. compute capability 8.9 (also supporting 9.0), with a version built against CUDA 11.8 and cuDNN 8.4.1.
- Added support for Nuke14.1 (which also is natively built against CUDA 11.8 and cuDNN 8.4.1)
- Added a check for GPU compatibility, i.e. disable the GPU processing options if the current GPU is not supported (instead of crashing).
- Improved the error logging. It's now much more clear if the node errors because you are running out of GPU memory.
- Added text information knobs to the "About" tab that prints out what CUDA/cuDNN versions the current build is against. Also what compute capability that is natively supported and what compute capability your current GPU supports and info about if you will or will not use JIT-compiled kernels. We've also added info about compatibility against Nuke's own AIR tools in relation to the current build configuration.
- Added a warning to the terminal during node creation if you are relying on JIT-compiled kernels. Just so you get some kind of heads up the first time the JIT-compilation kicks in (which usually takes just above half an hour). There are also warnings about the CUDA_CACHE_MAXSIZE environment variable if it's not set correctly. These warning can be suppressed with a new environment variable PIXELMANIA_SUPPRESS_KERNEL_WARNINGS if you want.
- Updated the NNFlowVector Utils gizmos "MotionVector_FrameBlend" and "MotionVector_FrameDistort" with a workaround that makes them work correctly again in Nuke13 and above.

v2.1.0 - June 2023

- Added support for dedicated render licenses.
- Added support for Nuke Indie.
- Added a new and modern model variant, available as the "AA" model in the variant drop down menu. This is a transformer based model which can handle more tricky situations better. It is using more VRAM on your GPU so you might need to lower the "max size" knob some to get it to run. You probably also want to turn on the new knob "separate inferences" (see below).
- Added a new knob called "separate inferences" which makes the node calculate the forward and backward vectors in two separate inference passes instead of at the same time. This makes the node use less VRAM on your GPU, with a slight render speed performance degration as a trade off.

v2.0.0 (official release) - March 2023
- Fixed a bug relating to when the bounding box of the input material isn't the same as the image format. This was a regression in the v2.0.0b5 release, and now it's working as it should again, exactly the same as in v1.5.1.

v2.0.0b5 (beta release) - January 2023
- Implemented matte input support, to be able to ignore a selected region when generating motion vectors. The output is instead a region filled and then machine learning enhanced area of vectors which represents what should have been there if the matted out object wasn't present when the material was filmed.
- Lots of restructuring of underlying code to make it possible to implement the matte support.
- Lots of optimisations and enhancements to make the plugin perform more stable, more memory efficient and a bit faster.

v1.5.1 - October 2022
- Patch release fixing a streaking error that occurred with the last processing patch (furthest to the bottom-right area of the processed image), in some resolutions (resolutions that needed padding to become dividable by 8).
- Improved the blending of the seams between processing patches. The problem was not always visible, but became apparent in some specific combinations of maxsize, overlap and padding values.

v1.5.0 - June 2022
- Fully re-trained the optical flow neural networks with optimized settings and pipeline. This results in even higher quality of generated vectors, especially for object edges/ silhouettes.
- To better handle high dynamic range material, all training has now internally been done in a logarithmic colorspace. Hence the "colorspace" knob became unnecessary and has been removed/deprecated.
- Implemented a "process scale" knob that controls in what resolution the vector calculations are happening in. A value of 0.5 will for example process the vectors in half res, and then scale them back to the original res automatically.
- Improved the user control of how many iterations the algorithm will do while calculating the vectors. The knob "iterations" is now an integer knob instead of a fixed drop down menu.
- Added a knob called "variant", to enable the user to choose between several differently trained variations of the optical flow network. All network variants produce pretty similar results, but some might perform better on a certain type of material. Hence we encourage you to test around. If you are unsure, go with the default variant of “A".
- Speed optimizations in general. According to our own internal testing, the plugin is now about 15% faster to render overall.
- Added an option for processing in mixed precision. This is using a bit less VRAM, and is quite a lot faster on some GPU architectures that are supporting it (RTX).
- Added an option for choosing what CUDA device ID to process on. This means you can pick what GPU to use if you got a workstation with multiple GPUs installed.
- Optimized the build of the neural network processing backend library. The plugin binary (shared library) is now a bit smaller and faster to load.

- Compiled the neural network processing backend with MKLDNN support, resulting in a vast improvement in rendering speed when using CPU only. According to our own testing it's sometimes using even less than 25% of the render time of v1.0.1, i.e. 4x the speed!
- Updated the NVIDIA cuDNN library to v8.0.5 for the CUDA10.1 build. This means we are fully matching what Nuke13.x is built against, which means our plugin can co-exists together with CopyCat nodes as well as other AIR nodes by Foundry.
- Compiled the neural network processing backend with PTX support, which means that GPUs with compute capability 8.0 and 8.6, i.e. Ampere cards, can now use the CUDA10.1 build if needed (see above). The only downside is that they have to JIT compile the CUDA kernels the first time they run the plugin. Please see the documentation earlier in this document for more information about setting the CUDA_CACHE_MAXSIZE environment variable.
- Internal checking that the bounding box doesn't change between frames (it's not supported having animated bboxes). Now it's throwing an error instead of crashing.
- Better error reporting to the terminal
- Added support for Nuke13.2

v1.0.1 - February 2022
- Initial release