

Table of Contents

1. Preamble
2. Installation
3. Usage and Knob Reference
4. NNFlowVectorWarp
5. Bundled utility tools
6. Frequently Asked Questions
7. Change Log

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 minimizing any impact on production.

1.2 Copyright

NNFlowVector User Guide, Copyright 2022 Pixelmania, a division of Brainspark Enterprises AB. All Rights Reserved.

The use of this User Guide and the NNFlowVector software is subject to an End User License Agreement (the “EULA”), the terms of which are included with the distribution of this software for reference. It is also available online at https://www.pixelmania.se/end-user-license-agreement. You also have to read and agree to all the Open Source Material terms, also included with the distribution of this software for reference, or detailed at https://pixelmania.se/open-source-material. This User Guide and the NNFlowVector software may be used or copied only in accordance with the terms of the EULA. This User Guide, the NNFlowVector software and all intellectual property rights relating thereto are and shall remain the sole property of Pixelmania.

Pixelmania assumes no responsibility or liability for any errors or inaccuracies that may appear in this User Guide and this User Guide is subject to change without notice. The content of this User Guide is furnished for informational use only.

Except as permitted by the EULA, no part of this User Guide may be reproduced, stored in a retrieval system or transmitted, in and form or by any means, electronic, mechanical, recording or otherwise, without the prior written consent of Pixelmania. To the extent that the EULA offers consent, such copies shall be reproduced with all copyright, trademark and other proprietary rights notices herein.

NNFlowVector compositing plugin, Copyright 2022 Pixelmania, a division of Brainspark Enterprises AB. All Rights Reserved.

NNFlowVectorWarp compositing plugin, Copyright 2026 Pixelmania, a division of Brainspark Enterprises AB. All Rights Reserved.

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}
}

@InProceedings{Morimitsu2025DPFlow,
  author    = {Morimitsu, Henrique and Zhu, Xiaobin and Cesar-Jr., Roberto M. and Ji, Xiangyang and Yin,
Xu-Cheng},
  booktitle = {The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)},
  title     = {{DPFlow}: Adaptive Optical Flow Estimation with a Dual-Pyramid Framework},
  year      = {2025},
}
```

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, Rocky8.x or Rocky9.x), or
- 64 bit Windows Operating System (for example Windows 10 Pro or Windows 11 Pro), or
- 64 bit macOS Operating System (with support for Apple Silicon M processors (arm64), for example macOS Sonoma 14.x)
- Nuke13.1 or higher on Linux, Nuke14.1 or higher on Windows, Nuke15.1 or higher on macOS, with a valid license.
  Nuke Indie is supported (to run the plugin in Nuke Indie, you have to use the latest version of Nuke14.x, Nuke15.x, Nuke16.x or Nuke17.x).

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 (anywhere on disk).

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 or **NNFlowVector.dylib** on macOS) 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

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

Below are examples of how to add the install path to the **NUKE_PATH** environment variable on different platforms:

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
You can easily edit this file by starting a new terminal and using the command:
nano .bashrc

macOS:
The simplest way to create this environment variable is to put a single line like this in your **.zshrc** file in your user’s home folder:

export NUKE_PATH=/full/path/to/plugin/install/folder
You can easily edit this file by starting a new terminal and using the command:
nano .zshrc

Windows:
In Windows 10/11, 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

2.3 Environment Variables

PIXELMANIA_NNFLOWVECTOR_DONT_RENDER_USING_GUI_LICENSES
The default behavior 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 behavior 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_NNFLOWVECTOR_FORCE_RENDER_USING_GUI_LICENSES
(This environment variable is new as of NNFlowVector v3.0.0.)
The default behavior of the node in batch/render mode is to look for dedicated NNFlowVector render licenses. If you want to force the plugin to directly checkout a GUI license instead of looking for a render license, please set the environment variable PIXELMANIA_NNFLOWVECTOR_FORCE_RENDER_USING_GUI_LICENSES to the value of “1”.
If you are relying on rendering using GUI licenses (**such as when you have a site license**), setting this variable will actually make the render startup process a bit faster because the node can directly checkout the correct license type. You will also get less log clutter in the license server because of directly checking out the correct license type.

PIXELMANIA_SUPPRESS_KERNEL_WARNINGS
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".

This variable does NOT apply to the macOS build (since the macOS build is accelerated by MPS instead of CUDA).

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, 8.6 or 8.9 GPU, i.e. the Ampere/Ada Lovelace 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 **4294967296** because that is 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 8.0/8.6/8.9 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/CUDA12.8 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 more modern releases of Nuke, i.e. Nuke14.1/15.0/15.1/15.2/16.0/16.1, 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 14.1/15.0/15.1/15.2/16.0/16.1 is hence using CUDA 11.8 and cuDNN 8.4.1 to fully be compatible with Nuke's own AIR tools.

When it comes to the newly released Nuke17.0, 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 RTX50xx series of NVIDIA GPUs. NNFlowVector for Nuke17.0 is hence using CUDA 12.8 and cuDNN 9.7.1 to fully be compatible with Nuke's own AIR tools.

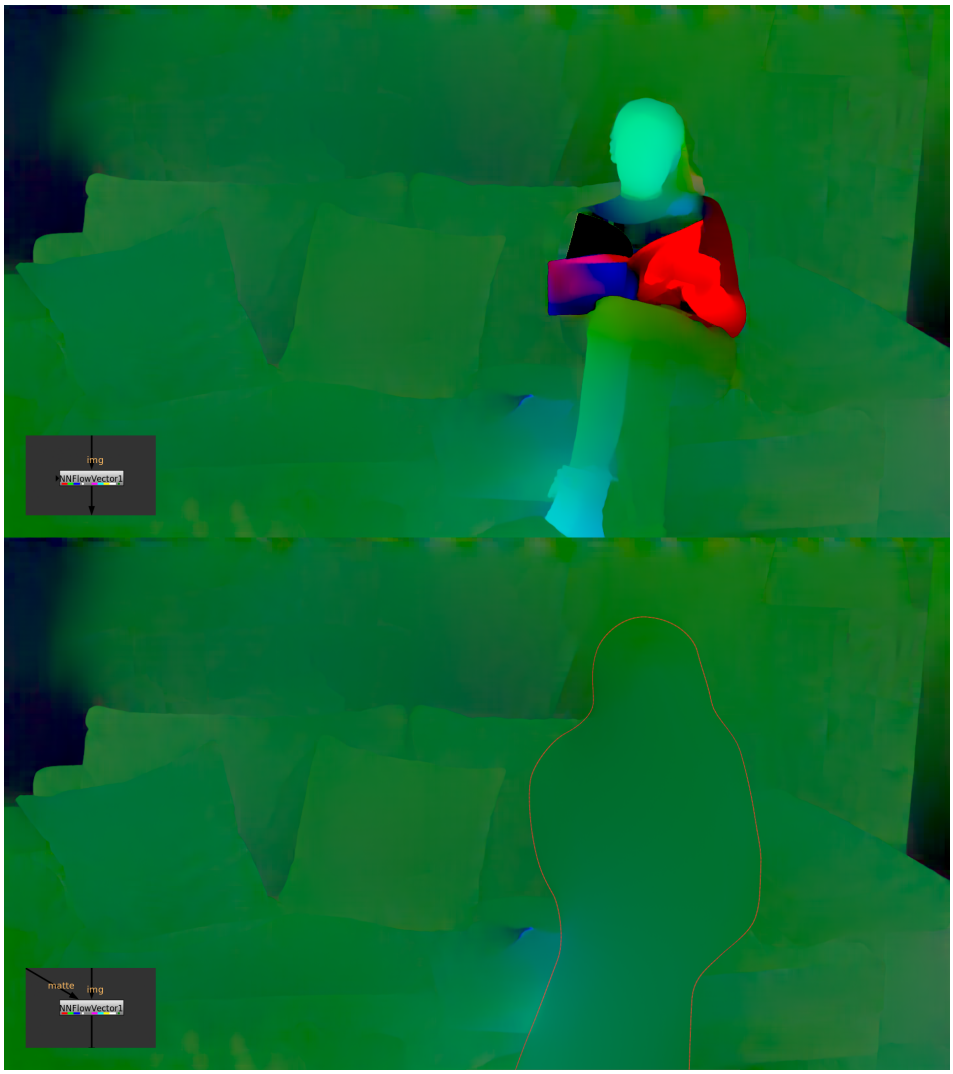
To see a chart of CUDA compatibilities with different NNFlowVector builds, please visit this page on our website: https://pixelmania.se/cuda-compatibility-chart/

This environment variable does NOT apply to the macOS build (since the macOS build is accelerated by MPS instead of CUDA).

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.
New in NNFlowVector v3.0 is that you can use the button “Create Write node” to get a Write node that is already set to the correct settings for the vectors NNFlowVector is set to generate.

SmartVector, and the tools that come bundled with it in NukeX, is truly a brilliant invention that opens up 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 v3.0: We are shipping our own new node for warping material using the produced vectors, called *NNFlowVectorWarp*. NNFlowVectorWarp is hence a direct alternative to using VectorDistort, and something we really do recommend to use instead when using vectors generated from the NNFlowVector node. The NNFlowVectorWarp node is directly developed for being compatible with and optimized for vectors generated by NNFlowVector, which will result in even cleaner and more stable tracks/warps/stabilizations. Please see the full documentation of NNFlowVectorWarp further down in this document.

Matte input
The 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 (the white area in the alpha channel defines the area to be removed). 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 optimize 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 optimize your output.

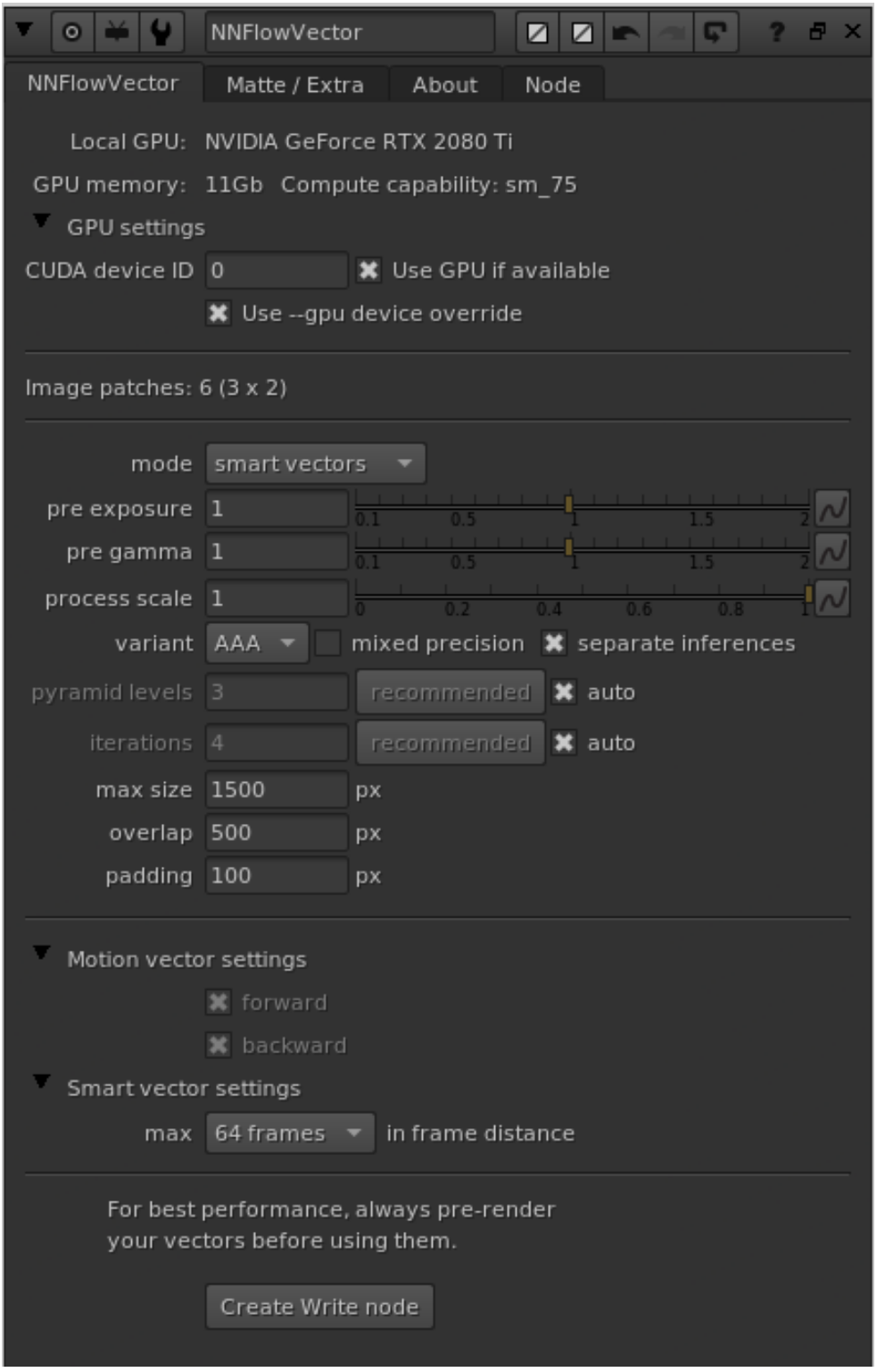
process scale

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

The variant (“A”, “AA” or “AAA”), sets what neural network training variant/model to use. **New in v3.0 is the great model “AAA”**. This is a big update to the older variants, and we do recommend to always run the “AAA” model. We were close to removing all other variants, but decided to keep the “AA” and “A” models for backwards compatibility/legacy’s sake for now (will probably be removed in a later release). The “AAA” model is similarly lightweight to the old “A” model (in terms of VRAM need), but produces way better vectors that are more detailed and less noisy (a lot smoother overall). It is also much better at handling larger input resolutions.

Note regarding NNFlowVector on macOS:
The “AA” variant is not available to be run using MPS acceleration on macOS, and the “Use GPU if available” knob will automatically be greyed out if you pick it. This is due to



some tensor functions not being MPS supported in the PyTorch implementation that comes bundled with Nuke. Please use the “AAA” variant instead.
Another note is that the old “A” variant is not able to be run on macOS in Nuke17.0 using “separate inferences” on, and hence that option will be grayed out in that specific case.

mixed precision
When this is “on" (which is not the default anymore) 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). In older versions of the node (v2.3.0 and lower) we recommended to have this turned “on”. But the new “AAA” variant is not behaving as well in mixed precision, and hence we recommend to not have “mixed precision” turned on (and we have turned it off in the default settings as well).

separate inferences
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. When “separate inferences” is set to “off”, the node calculate the forward and backward vectors in parallel using one inference run through the neural network (a touch faster, but uses more memory). The default setting for this knob is now set to “on” (as of v3.0).

pyramid levels
This knob is new in v3.0, and is connected to the new variant “AAA”. It defines how many pyramid levels that the model uses internally in its model structure. This knob is disabled when using other variants than “AAA”.
You also have a button with the label “recommended” next to the pyramid levels knob. This can be pressed to calculate and set the recommended value of pyramid levels which is related to the input resolution.
There is also an “auto” checkbox, which disables the “pyramid levels” knob and the “recommended” button, and instead directly uses the live calculated recommended value internally.

iterations
Defines how many refinement iterations the solve algorithm will do for the chosen model variant. This only applies to the variants “AAA” and “A”, and is hence greyed out if you use the “AA” variant.
Similarly to the workings of “pyramid levels”, there is a “recommended” button and an “auto” checkbox connected to the “iterations” as well. Good to know is that the recommended value for “iterations" is very different for the different variants. For the “AAA” variant we recommend setting iterations to around 4, and for the “A” variant we recommend setting iterations to around 25.

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 1500, which means that the max dimensions an input patch would be allowed to be is 1500×1500 pixels. From our experience that will use up to around 8Gb of VRAM on your graphics card (applies to the AAA/A variants. For the “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 500.

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 100. 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. The same is true for the NNFlowVectorWarp node.

smart vector mode: frame distance
The drop down menu specifies the highest value of "frame distance" that you can use in a NNFlowVectorWarp / VectorDistort node when using the rendered SmartVector compatible output of NNFlowVector. If you want to have all the possible options available in a NNFlowVectorWarp 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 optimize 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).

Create Write node
This is a convenience button that will create a pre-configured Write node for easy pre-rendering of the vectors (which is always the recommended workflow). Please note that the settings are different depending on if you are generating motion vectors or smart vectors. Motion vectors use interleaved channels, while smart vectors use multi-part EXRs.

CUDA device ID
This knob is 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.

This knob does NOT exist in the macOS build (since the macOS build is accelerated by MPS instead of CUDA).

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.

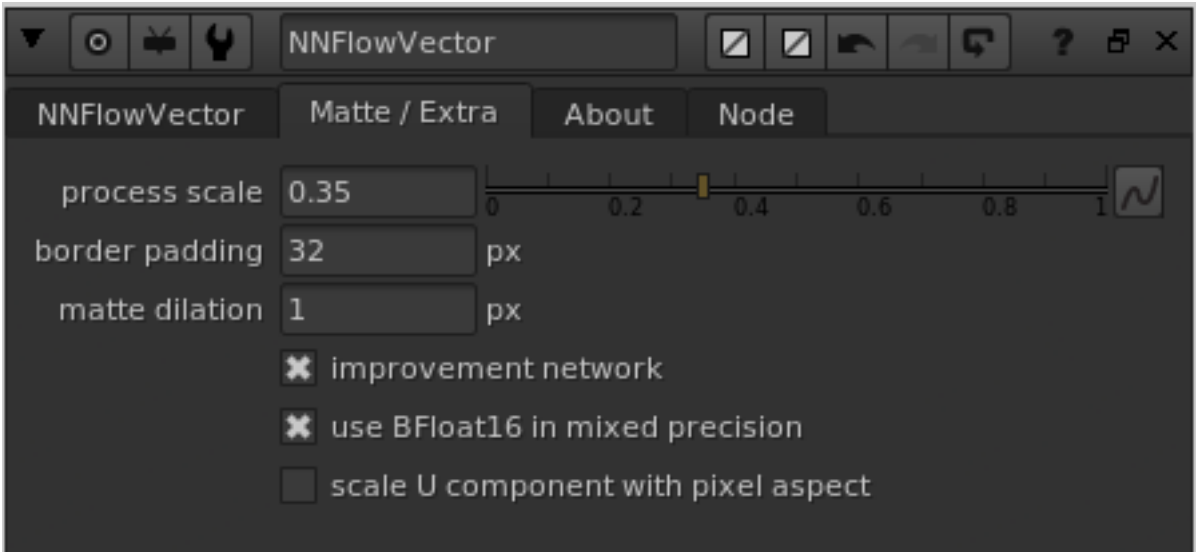
We highly recommend always having this knob "on", since the algorithm will run A LOT faster on the GPU than on the CPU. To give you an idea of the difference, we've seen calculation times of the same input image be around 2.0 seconds/frame using the GPU and about 34 seconds/frame on the CPU (a difference 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.

Use --gpu device override
This is a new knob in v3.0, and is there to help with batch rendering on systems with multiple GPUs. When this knob is set to "on" (which is the default), then node will not blindly use the device ID set in the knob "CUDA device ID" but will rather adhere to the --gpu command line parameter (from the command that started the nuke batch render). This means that NNFlowVector will behave like Foundry's own GPU accelerated nodes, and render using the specified GPU in the Nuke batch render command. This enables you to have a setup in the GUI, when you are working with the node as an artist, but still be able to farm the exact same Nuke script on different boxes with different GPU configurations.

The "Matte / Extra" 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 (the white regions in the alpha, i.e. values of 1.0, are the ones that are getting inpainted). 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 0.35.

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 artifacts 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 32 pixels.

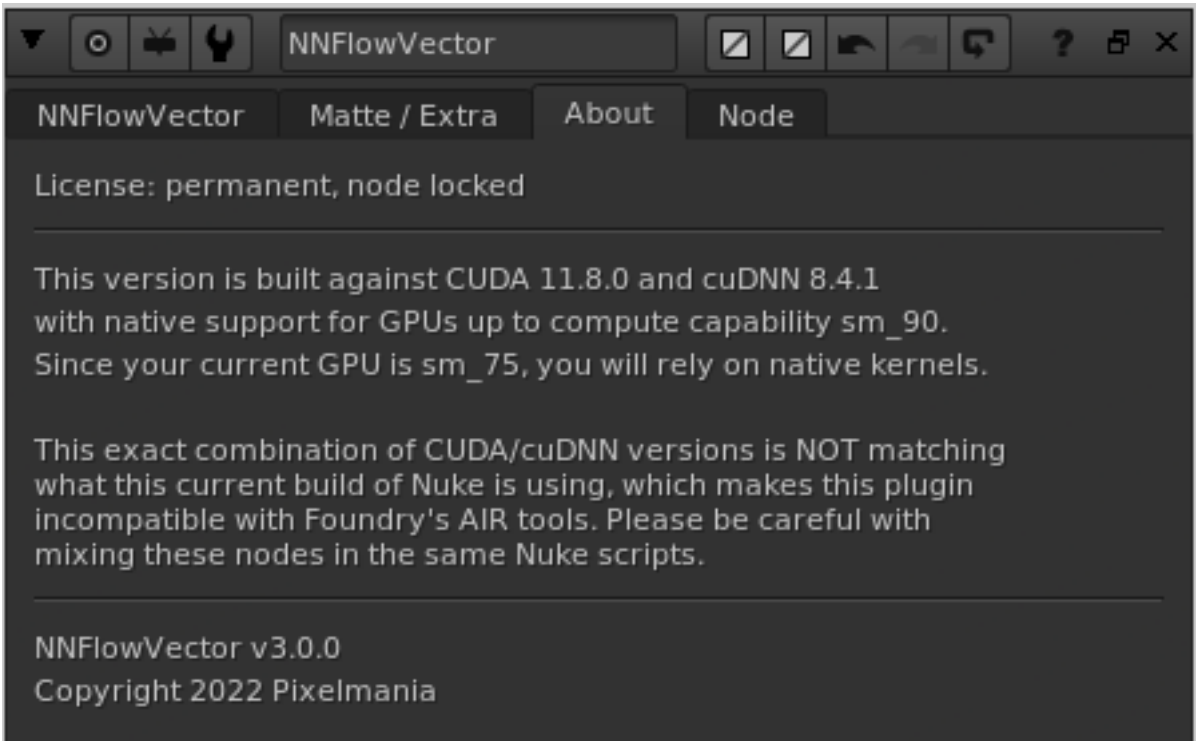


matte dilation
The matte dilation setting is specifying how many pixels to dilate the matte before it's used for the inpainting and improvement neural network. Because the original matte is used to actually keymix in the calculated vectors in the matte region, this dilation setting makes the end result 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 artifacts in some cases. Keep this value as low as possible, even set it to 0, as long as you're not experiencing the edge artifacts. If you get them, slowly increase this setting. The default value is 1 pixel.

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.

Use BFloat16 in mixed precision
If this knob is "on", it will change the data type used in mixed precision processing from classic half floats (16bit) to bfloat16 (if your GPU is modern enough to support it). This is usually better and you should keep this on (default).

scale U component with pixel aspect
This knob specifies if the pixel aspect should be multiplied into the U component of the output vectors or not. This is only relevant when working with anamorphic material (i.e. non-square pixel aspects). The default is "off" which is usually what you want, but we expose this setting to handle cases where it's needed.



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. It 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. NNFlowVectorWarp



One of the big news in v3.0 is the addition of the node **NNFlowVectorWarp**. This is our own take on a node with similar abilities to NukeX’s VectorDistort, but being tailor made and optimized for the vectors produced by the NNFlowVector node (especially the new “AAA” model variant). **With the NNFlowVectorWarp node you can track/warp patches onto live**

action footage, you can stabilize live action footage as well as warp stabilized footage back to it's original motion. You can also output STMaps of the warps produced, to be able to apply the warps in a very efficient and optimized way using the classic STMap node (part of Nuke's default toolset). We always recommend pre-rendering both the vectors from the NNFlowVector node and also the STMaps produced from the NNFlowVectorWarp node, to depend on fast disk read and fast/lean warp algorithms instead of depending on these very heavy calculations in a live comp environment.

4.1 Usage of the Node

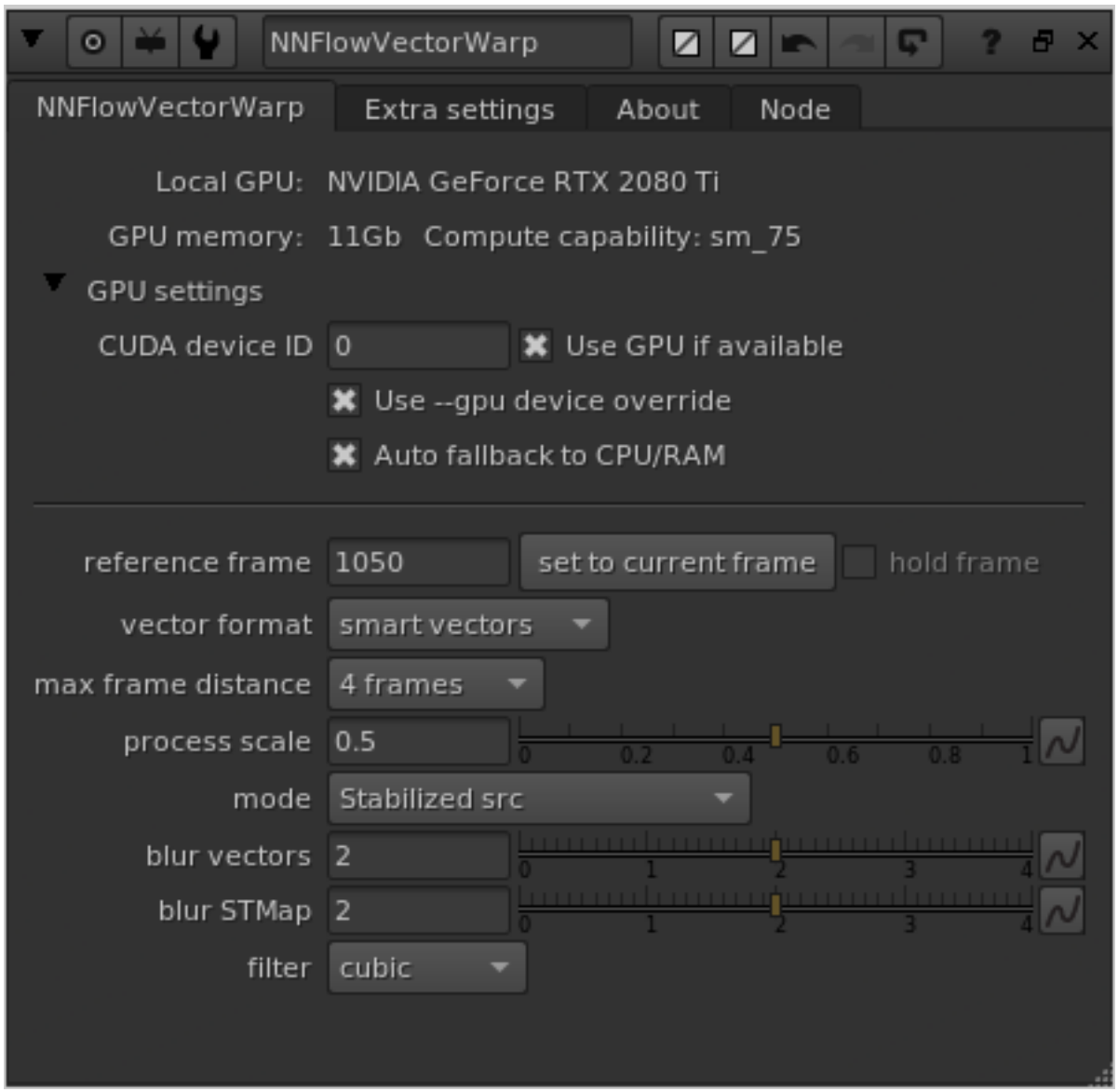
The way the NNFlowVectorWarp node works is that you feed it with either normal "motion vectors" or "smart vectors", which you input to the node's input named "vectors". You then set a "reference frame" to what frame you want it to lock onto. The reference frame is the frame that will be non-deformed that the rest of the time line will reference.
There are two different modes you can use, either "Warp" or "Stabilize" (you can think of it as "forward warp" and "inverse warp" if you like). Both these modes either work on provided image material, or produces STMaps directly (for pre-rendering and later use with the classic STMap node in Nuke). If you produce STMaps directly, you don't need to connect anything to the "img" input. If you instead choose "Warped src" or "Stabilized src" it will apply the warps directly onto the material provided into the "img" input. There is a neat special case where, if you use the mode "Stabilized src", and don't connect anything to the "img" input, it will instead use both the vectors AND the RGB(A) channels from the "vector" input. This is very nice when you want to stabilize a sequence that you already pre-rendered from NNFlowVector, because that node by default also outputs the beauty RGB(A) channels. This way you can have a very slick and simple processing comp script as such:
Read > NNFlowVector > Write (channels: all) > Read > NNFlowVectorWarp (only connected using the "vector" input, and set to "Stabilize src") > Write (fully stabilized plate)

4.2 Knob reference

The following are the available user knobs on the NNFlowVectorWarp node.

reference frame
The reference frame is the most important knob on the node, hence being the first. This sets which frame should be treated as the hero reference, i.e. the frame that should not be deformed/distorted/warped but that every other frame on the timeline references/relates to. There is a convenience button next to this knob called "set to current frame" which does exactly that. Just press it to set the current frame as the value of the "reference frame" instead of typing it manually. The node also defaults to setting the "reference frame" to the current frame when the node is created.

hold frame
The checkbox "hold frame" is only relevant when using the mode "Warped src". If it's "on" it basically acts as a FrameHold node on the "img" input, set to the same frame as the "reference frame". The default is "off" because quite often you want the full sequence, that is being input to the "img" input, to be warped. But it's good to have the convenience option available.



vector format
The vector format knob specifies what format of vectors that are input into the “vectors” input. Basically it defines if the node should work using classic “motion vectors” or if it should work using “smart vectors”. Set it to whatever you have set the upstream NNFlowVector node to. The two options work similarly, but not exactly the same. The “motion vectors” are vectors that only describe the movement from one frame to another (both forward and backward in time). “Smart vectors” on the other hand describe the same things as motion vectors, but then in addition it describes bigger jumps between frames as well, with distances being 2, 4, 8, 16, 32 and 64 frames. The smart vectors enables the warp algorithm to optimize heavily and not having to do as many iterative warping steps as with motion vectors. This also makes it read less frames of vectors from disk, so the I/O becomes faster as well. If you can, please do experiment with “smart vectors” in favor of “motion vectors”, and test different “max frame distance” steps, like for example 2, 4 or 8.

max frame distance
This is related to the information above (the “vector format” knob). Basically this knob sets how big “frame distance” jumps it is allowed to do when using “smart vectors” as the vector input. You also have to provide the node with matching frame distances, i.e. if you want to

use “max frame distance” 16 you also have to provide the node with a pre-render of vectors that contain “frame distances” up to 16. If you set the “max frame distance” to 16, and you only provide it with vectors that supports up to frame distance 4, for example, it will automatically fall back to frame distance 4 instead (without erroring).

process scale
The process scale sets the internal “working resolution” as a scale factor of the input resolution. If you, for example, feed the node with UHD (3840x2160) and set the process scale knob to 0.5 (which is the default), the node will calculate all the warping internally in full HD (1920x1080) and then scale the resulting STMap back up to full resolution in the end. If you are using a mode that is producing warped images as output instead of the STMaps, it’s good to know that all the warping always happens in full resolution. Basically it’s only the calculation of the internal STMap that is done at the process scale resolution. The benefit of not always calculating the STMap in full resolution is twofold, more specifically speed and less memory usage. Storing the internal vectors when process scale is set to 0.5 is only using 25% of the memory, and the calculations goes a lot faster. In most common cases, you are not even loosing important details, but is only reaping the benefits of the improved performance. Please see the “blur vectors” and “blur STMap” knob descriptions below for some related information.

mode
The mode is probably the second most important knob after the reference frame. This sets what the output of the node should be. Are we forward warping or are we stabilizing? Are we processing footage that is input using the “img" input, or are we producing STMaps to be used later on? The four different modes are the following:
- **Warped src** - Warping the footage provided using the “img” input
- **STMap (warp)** - Producing STMaps for forward warping
- **Stabilized src** - Stabilizing the footage provided using the “img” input. If no input is provided, the node will instead use the RGB(A) channels from the “vector” input for your convenience.
- **STMap inverse (stabilize)** - Producing STMaps for stabilizing

blur vectors
This is a pre-blur of the provided vectors to be able to easily filter too high frequency vectors. Basically you can get rid of some small jitter, without trading overall tracking precision, if you do blur the input vectors slightly. Usually a small value between 0 and 4 is good, but what value to use is very depending on what material you are processing of course. If you are unsure, just use the default value 2.

blur STMap
This is a post-blur of the resulting/calculated internal STMap, but after it has been scaled up to be full resolution again. If the process scale was smaller than 1.0, it is usually a good idea to do a bit of blurring of the STMap, just to slightly smooth the upscale filtering artifacts. Do experiment with values between 0 and 4 to find what best fits your material. The default is set to 2.

filter
This setting is only relevant when using the modes for processing footage, i.e. “Warped src” and “Stabilized src”. It is defining what filter to use for the warping of the material when applying the internal resulting STMap. This will produce the exact same result as if you

instead output the STMap, and then later using a classic STMap node in Nuke and set the filter knob in that node to the same value as this filter knob.

CUDA device ID
This knob is 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.

This knob does NOT exist in the macOS build (since the macOS build is accelerated by MPS instead of CUDA).

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 output of the node, 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.

Use --gpu device override
This is a new knob in v3.0, and is there to help with batch rendering on systems with multiple GPUs. When this knob is set to "on" (which is the default), then node will not blindly use the device ID set in the knob "CUDA device ID" but will rather adhere to the --gpu command line parameter. This means that NNFlowVector will behave like Foundry's own GPU accelerated nodes, and render using the specified GPU in the Nuke batch render command. This enables you to have a setup in the GUI, when you are working with the node as an artist, but still be able to farm the exact same Nuke script on different boxes with different GPU configurations.

Auto fallback to CPU/RAM
When this is "on" (default) together with when "Use GPU if available" is "on", the node will check if the needed vectors can fit in the currently available GPU VRAM. If they can't fit, the node will automatically fallback to processing on the CPU and normal RAM instead of erroring. This could be the preferred way, if you are prioritizing getting a result (even if the processing might take longer). It could also not be the preferred way, if you rather have the node erroring in such a case, for example when batch rendering in a farm environment. You might want it to error, so another machine with a larger GPU can pick up the frame and process it instead. The option is there for you to use accordingly.

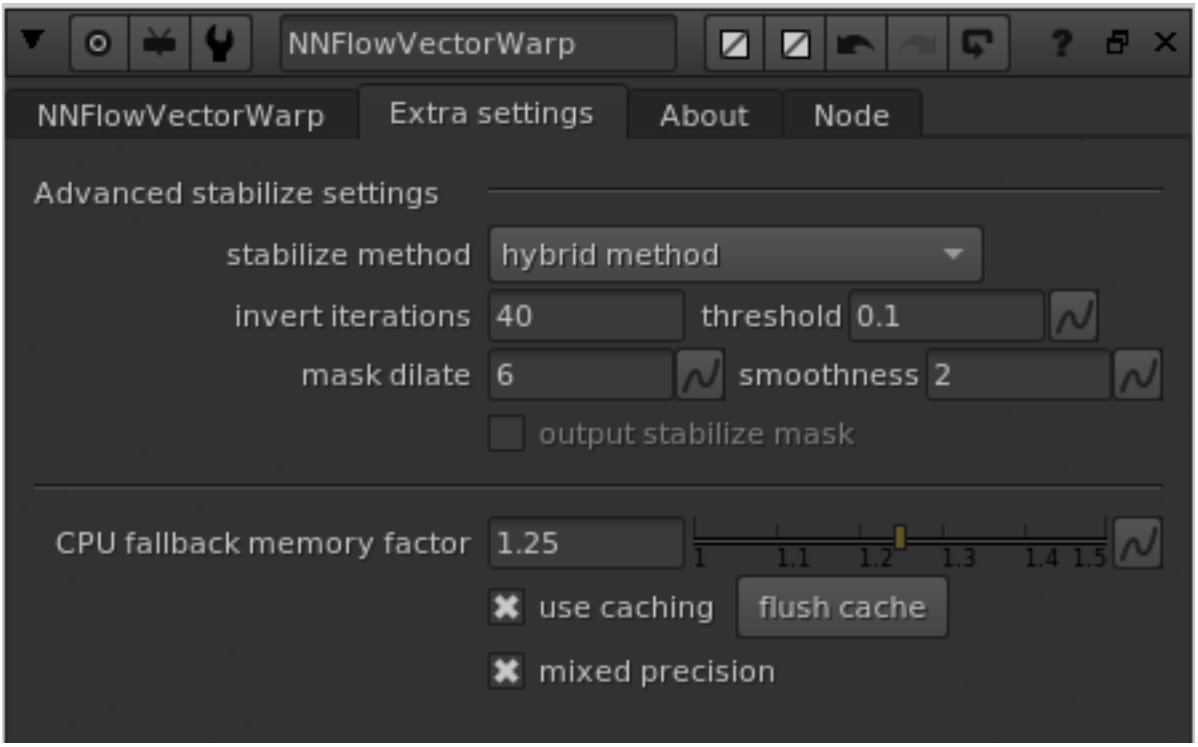
Worth noting is that half of the processing might be accelerated using the GPU, and the other half using the CPU/RAM. This can happen if you are stabilizing and using the "hybrid method" which basically uses both forward and backwards vectors at the same time. It could then be that the forward vectors can fit into the GPU VRAM, but after they have transferred, the backward vectors can't fit and has to stay on the CPU/RAM path.

Please note that there is another related knob called “CPU fallback memory factor” that is documented a bit further down (since it’s in the “Matte / Extra” tab).

The “Extra settings” tab’s knobs:

stabilize method
This knob is part of the “Advanced stabilize settings” group, which means it’s only related to when you are using stabilization as the mode. This drop down menu has three options;

- **prio round-trip/reversability**: This mode stabilizes the material, but prioritizes being able to warp it back to the original as close as possible.
- **prio reference matching**: This mode stabilizes the material, but prioritizes matching the reference frame as closely as possible.
- **hybrid method**: This mode is the recommended mode (default), and is doing an automatic split between both of the other methods. Basically best of two worlds. You should very rarely need to change from using “hybrid method”.



invert iterations
This knob is part of the “Advanced stabilize settings” group, which means it’s only related to when you are using stabilization as the mode. In addition, this setting is only related to when you have set “stabilize method” to either “prio round-trip/reversability” or “hybrid method”. This knob is for tweaking the number of iterations in an internal solver for the round-trip convergence. You will probably never need to change it from the default value of 40. (We might even hide this knob in a future release).

threshold
This knob is part of the “Advanced stabilize settings” group, which means it’s only related to when you are using stabilization as the mode. In addition, this setting is only related to

when you have set “stabilize method” to “hybrid method”. The threshold is defining an internal error threshold that is used for automatically creating the split keymix alpha that is used by the “hybrid mode” to mix between the other two modes. The default value of 0.1 should be fine for most situations.

mask dilate
This knob is part of the “Advanced stabilize settings” group, which means it’s only related to when you are using stabilization as the mode. In addition, this setting is only related to when you have set “stabilize method” to “hybrid method”. The mask dilate knob sets how many pixels the calculated split keymix alpha is dilated before it’s used.

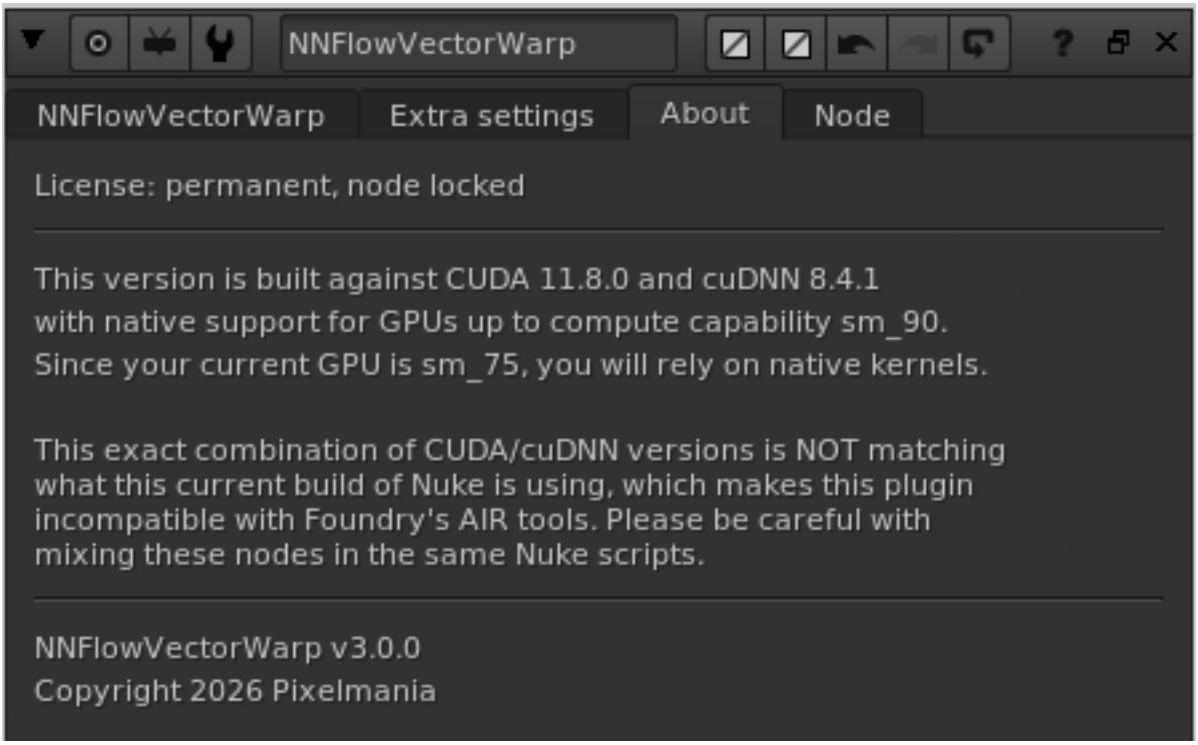
smoothness
This knob is part of the “Advanced stabilize settings” group, which means it’s only related to when you are using stabilization as the mode. In addition, this setting is only related to when you have set “stabilize method” to “hybrid method”. The smoothness knob sets how much the calculated split keymix alpha is blurred before it’s used. The default value of 2 should be fine in most cases.

output stabilize mask
This knob is part of the “Advanced stabilize settings” group, which means it’s only related to when you are using stabilization as the mode. This knob is specifically only enabled when you output a stabilization STMap from the node, and the “stabilize method” above is set to “hybrid method”. When the checkbox is set to “on” the node will use the B channel of the output STMap (which is an unused channel in general) to fill it with the split keymix alpha that is used for the hybrid mode to mix the other two different methods. The reason this option exists is so you can get a visualization of what the other knobs above actually do to the split keymix alpha. Since the B channel is not used when warping with an STMap, it doesn’t matter if you leave this “on” or “off”.

CPU fallback memory factor
This knob is related to the “Auto fallback to CPU/RAM” knob described above. If that knob is “on”, this “CPU fallback memory factor” knob is enabled. Here you can set how much safety margin you want the node to use when deciding if the retrieved vectors should be processed on the GPU or not. The default value of 1.25 basically means that it needs to exist 25% extra headroom of available GPU VRAM if the vectors and the following calculations should be done using the accelerated GPU path, otherwise the processing will happen using the slower CPU and normal RAM path.

use caching
This knob turn the internal caching “on” or “off”. The default is “on” which we very much recommend having. The cache saves as much intermediate vector data that it can reuse as possible. Quite often you are jumping between consecutive frames, and the speedup the use of a cache is making in such a case is huge.

flush cache
You should never need to flush the cache manually because the cache handling internally in the node is fully automatic. But we leave this button here if the need suddenly arise.



mixed precision
When mixed precision is “on” (default), the node will use half floats for operations that are supported. This makes the node use less memory and run faster, with a tiny tradeoff in accuracy. We definitely recommend having this “on” for the NNFlowVectorWarp node.

The “About” tab:

There is an additional tab called “About” in the knob panel for NNFlowVectorWarp. 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)

5. Bundled utility tools

5.1 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.

5.2 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 be 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 neighboring 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 neighboring 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 neighboring 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.

6. 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 Nuke14.x, Nuke15.x, Nuke16.x or Nuke17.x.

How are licenses consumed?

All of Pixelmania’s licenses are per host, i.e. you can have multiple jobs using NNFlowVector/NNFlowVectorWarp 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 behavior 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 behavior 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”).
To instead force the node to use GUI licenses when rendering, you can set the environment variable PIXELMANIA_NNFLOWVECTOR_FORCE_RENDER_USING_GUI_LICENSES to the value of “1”. This can be good in a farm environment and you are having a site license. (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):

- 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)**
- 12.0 (Blackwell)***

* 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, CUDA12.8)**. The CUDA11.1/CUDA11.2/CUDA11.8/CUDA12.8 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, CUDA12.8)**. The CUDA11.8/CUDA12.8 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.

*** The Blackwell architecture, i.e. the NVIDIA RTX50xx series of cards, works with all CUDA variants of NNFlowVector (**CUDA10.1, CUDA11.1, CUDA11.2, CUDA11.8, CUDA12.8)**. The CUDA12.8 version got native support and will work directly. The CUDA10.1/CUDA11.1/CUDA11.2/CUDA11.8 versions will need to compile the CUDA kernels from the PTX code. Please see more info above in the environment variables section.

The above section does NOT apply to the macOS build (since the macOS build is accelerated by MPS instead of CUDA).

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.
Since the release of Nuke14.1/15.0/15.1/16.0/16.1/17.0 our plugins are linked directly against the bundled PyTorch version that is released together with Nuke. This makes the binaries much smaller than before, even up to around 2Gb smaller.

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/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.

7. Change Log

v3.0.1 - October 2026
- Added support for Nuke17.1 on all platforms (Linux, Windows and macOS). Since nothing else has been changed, we are not versioning up, but just releasing the new builds.

v3.0.1 - March 2026
- Fixed a bug that sometimes made the smart vector layers for frame distances 16, 32 and 64 to be corrupt when batch rendered from the command line.

v3.0.0 - March 2026
- Added the new model variant “AAA”.
- Added the new node NNFlowVectorWarp (similar to Nuke’s VectorDistort but optimized for the vectors from NNFlowVector).
- Added support for Nuke16.1 and Nuke17.0 on all platforms (Linux, Windows and macOS).
- Added the functionality to automatically process on the GPU device ID that was specified in the Nuke batch render command (set by the --gpu parameter).
- Added the environment variable PIXELMANIA_NNFLOWVECTOR_FORCE_RENDER_USING_GUI_LICENSES

v2.3.0 - March 2025
- Added support for Nuke15.2 and Nuke16.0 on all platforms (Linux, Windows and macOS). Since nothing else has been changed, we are not versioning up, but just releasing the new builds.

v2.3.0 - July 2024
- Added support for Nuke15.1 on all platforms (Linux, Windows and macOS).
- Added a new transformer variant called “BB”.
- Bug fixed an issue where the motion vectors were not generated correctly for anamorphic material.

v2.2.1 - April 2024
- Added support for Nuke15.0 on macOS for Apple Silicon M processors (arm64), accelerated using MPS (Metal Performance Shaders).

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 degradation 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 optimizations 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