We thank all reviewers for their valuable comments. Below, we address their main concerns by quoting the comment followed by our response.

R2: Q1. Use the same architecture for all datasets: We indeed did this. To be precise, we only performed the architecture search once on the SceneFlow dataset and fine-tuned the weights on each benchmark separately. This implies the generalization capability of our proposal to a great degree. We conjecture that the main reason here is the use of a refined search space in our algorithm. In words, rather than growing the search space blindly in the hope of finding a good architecture, we have used the task-specific physics and inductive bias to constrain the search space.

R2: Q2. Similar approach in different domains: We agree that our design is tailored for the task of stereo matching and cannot be considered as a domain agnostic solution for a different problem. For different domains, better task-dependent NAS mechanisms are still a suitable solution to efficiently incorporate inductive bias and physics of the problem into the search space. We will reflect this in a revised version of our paper.

R2: Q3. Interpretability: We will tighten up our language based on your comment.

R2: Q4. Better performance on large error thresholds: We agree with the reviewer that the downsampling operations might prohibit the network to learn sub-pixel accuracy. It might also because of the loss function that does not encourage sub-pixel accuracy. We will acknowledge the issue (Bad 1.0 and 2.0 errors) and discuss accordingly in a revised version of our paper.

R4: Q1. Technical Contribution: Our method is the first NAS based method that can successfully do a *full architecture* search for an end-to-end stereo matching network. Note that directly applying [17] to stereo matching for a full architecture search is not viable (due to the huge memory requirements for high-resolution dense predictions, it can only search networks with limited layers). Also, as acknowledged by other reviewers, NAS has shown great success in classification tasks while not been very effective for dense prediction tasks yet. In our paper, we have successfully demonstrated that our NAS methods can achieve better performance than human-designed architectures by ranking 1 on various stereo matching benchmarks. We will revise the text to make this more clear. We will also release our code to ensure the reproducibility of the work and to improve this field.

R5: Q1. Formula for updating β: The parameter β is updated similar in spirit to that of α. Specifically, the following formula is used to update β, where k represents the downsampling rate.

$$\boldsymbol{s}_l = \beta^l_{\frac{k}{2}\to k}\mathcal{C}(\boldsymbol{s}_{\frac{k}{2},l-1}, \boldsymbol{s}_{k,l-2}; \alpha) + \beta^l_{k\to k}\mathcal{C}(\boldsymbol{s}_{k,l-1}, \boldsymbol{s}_{k,l-2}; \alpha) + \beta^l_{2k\to k}\mathcal{C}(\boldsymbol{s}_{2k,l-1}, \boldsymbol{s}_{k,l-2}; \alpha), \tag{1}$$

$$\beta^l_{\frac{k}{2}\to k} + \beta^l_{k\to k} + \beta^l_{2k\to k} = 1 \quad \text{and} \quad \beta^l \geq 0, \ \ \forall l, k. \tag{2}$$

R5: Q2. Compare with AANet: AANet(CVPR20) was officially published in June, 2020, after the deadline of NeurIPS. Structure-wise, the difference between our solution and AANet is that AANet builds multiple multi-scale cost volumes and processes them with 2D convolutions while our method constructs a feature volume and processes it with 3D convolutions. Our method benefits from fewer parameters (1.8M *vs* 3.9M) while enjoying higher performances (KITTI12 1.45% *vs* 2.04%, KITTI15 1.65% *vs* 2.03%, Middlebury 2.75% *vs* 10.8%). Per R5's comment, we will include AANet in a revised version of our paper.

R6: Q1. Takeaways from the found architecture: 1. The feature net does not need to be too deep to achieve good performance; 2. Larger feature volumes lead to better performance ($1/3$ is better than $1/6$); 3. A cost volume of $1/6$ resolution seems proper for good performance; 4. Multi-scale fusion seems important for computing matching costs (*i.e.*using a DAG to fuse multi-scale information). We will add a discussion about it in a revised version of our paper.

R6: Q2. Our method *vs* AutoDispNet: AutoDispNet has a very different network design philosophy than ours. It is a large U-Net-like architecture and tries to directly regress disparity maps from input images in pixel space (in contrast to our design which benefits from a feature and matching networks). Table on the right provides a head-to-head comparison. We will include this along a discussion in a revised version of our work.

	Search Level	Params	KITTI 2012	KITTI 2015	Runtime
AutoDispNet	Cell	111M	1.70%	2.18%	0.9 s
LEAStereo	Full Network	1.8M	1.13%	1.65%	0.3 s

Table 1: AutoDispNet *vs* LEAStereo

R6: Q3. Test Middlebury on quarter resolution: We would have liked to test Middlebury on a quarter resolution, but due to the strict submission policy, every method can only be submitted once. We are in the middle of negotiating the issue with the organizers of the Middlebury challenge to see if we can have another submission at the time of writing the rebuttal. Once we have it, we will add the results to the revised paper.

R6: Q4. Minor implementation details: A similar choice was also considered in other papers (*e.g.*, GA-Net). In comparison to $1/4$, the downsampling to $1/3$ will remove the need of upsampling twice. For question (b) and (c), please refer to the first question of R2.