We thank all the reviewers for their time and effort. We appreciate the constructive feedback as well as the acknowledgment of the significance of this work in the review reports. Let us summarize the main contributions of this work as 1) extending the well known Robust PCA denoising technique to the manifold setting thus greatly broadened the applications and 2) providing a solid theoretical guarantee for the method. The key observation is that the success of the proposed method depends solely on the intrinsic property of the data manifold instead of specific sampling procedures (Theorem 4.2), which makes our extension non-trivial. Last but not least, to avoid the hassle of choosing tuning parameters, we proposed a curvature estimation method that could be useful in other contexts.

We are particularly grateful for the suggestion of the reviewers about Section 5-6. We will restructure these two sections for clarity. Specifically, we will move Sect. 5.1 (A short review of related concepts in Riemannian geometry) to the appendix, and use the released space to better explain the curvature estimation idea (e.g., add derivation of Eq. (12), add explanations of the parameters in Figure 1 and their relation to those defined in the context of Sect. 5.2 and Sect. 5.3, and summarize the curvature estimation procedures in a small algorithm). We will also follow Reviewer 1's and Reviewer 2's advice to add the derivation of Eq. (13) and Eq. (15).

Due to space limitations, below we only address the major concerns raised by the reviewers.

About the curvature estimation method: we apologize for not including enough details in the description of the proposed curvature estimation method. We agree with Reviewer 2 that considering its importance, we should make room to better explain this idea. Generally speaking, there are indeed parameters to be set in the curvature estimation step, but the main algorithm (Algorithm 1) is rather *insensitive* to the choice of these parameters. Specifically, in Sect. 5.2, we explained how to estimate the average curvature at each data point (Eq. (8)) which is used later to set the parameter λ_i in the NRPCA formulation (for completeness, we also mentioned how the same idea can be used to estimate the overall curvature of manifold in Line 171-174, but the overall curvature is *not* used in the proposed method). When estimating the curvature at some point p, our method requires (Sect. 5.2) choosing n other points independently and uniformly at random from a neighborhood of p (say the neighborhood has a radius r_1), compute the curvatures of the geodesic curves joining p and these n neighboring points using Eq. (7), and then take the average of the computed curvatures to derive Eq. (8). During this process, we need to pre-set the aforementioned parameters r_1 and n, as well as the size r_2 of the kNN in the Dijkstra's algorithm used to compute the geodesic distances (mentioned in Line 165). However, through numerical experiments, we found that the final result of the main algorithm (Algorithm 1) was very robust to different choices of all these parameters. We will include in the paper this remark as well as some numerical experiments to justify the claimed stability of Algorithm 1 w.r.t. the choices of parameters. We are also able to theoretically justify this approach under the ideal uniform sampling assumption.

About kNN and ϵ-neighborhood: we thank Reviewer 1 for raising this issue. kNN is used in the actual implementation of the proposed algorithm while ϵ-neighborhood is used in establishing Theorem 4.2 (Line 105). This is a common practice in manifold learning (e.g., in the proof of the convergence of the graph Laplacian to the Laplace-Beltrami operator of the manifold [1]), as the mathematical treatment for the ϵ-neighborhood is much easier than kNN, while the implementation of kNN is more stable than ϵ-neighborhood. More importantly, the performance of these two are similar to each other under the uniform and sufficient sampling assumption.

Does the methodology presented in Section 2 work for non-Gaussian noise too? Answer: The theoretical result does not rely on the distribution of noise, so the proposed method also works when noise is non-Gaussian, as long as its magnitude is still small.

What is the dimension of the NRPCA space, is it two? Answer: Similar to Robust PCA, the output (i.e., the denoised data matrix) of NRPCA is of the same size as the input (i.e., the noisy data matrix). That is to say, NRPCA does not reduce the dimension of the data and is a pure denoising technique.

Classification results based on LLE and Isomap: In the numerical experiments, we conducted LLE and Isomap on the NRPCA denoised data to see how the denoising affects the performances of LLE and Isomap. We used LLE and Isomap instead of tSNE because they are much more sensitive to outliers and Gaussian noise, thus are good indicators of whether or not the noises are successfully removed. As recommended by reviewer 1, we implemented a classification task on digits 4 and 9 from MNIST dataset, based on the denoised data in the numerical section. We first applied Isomap to the denoised data, then used SVM with Gaussian kernel for classification. The cross-validated classification error rate is 6.75%, while the error rate is 22.30% when applying SVM to the 2D data embedded by Isomap from the original noisy data, which indicates that our method effectively removed the noise, and the dimension reduction under Isomap became better.

Is the neighborhood patch defined with respect to X_i or $\tilde{X}_i$? What happens for $T \gg 2$? Answer: the initial neighborhood patch is defined w.r.t. the noisy data. For $T > 1$, patches are updated along with the variables, and as $T \to \infty$ converges to (hopefully) the neighborhood patches corresponding to the clean data. In our experiment, we did see a trend of convergence every time when T gets larger. The reason we only choose $T = 1, T = 2$ in the figures is that the results do not change much after $T > 2$.

[1] Hein et al., "*From graphs to manifolds–weak and strong pointwise consistency of graph Laplacians*," 2005.