# Sparse Iterative Closest Point

Paul Caucheteux

March 22, 2024

**Abstract**

This work provides an analysis and a replication of the Sparse Iterative Closest Point algorithm introduced in [BTP13] by Sofien Bouaziz, Andrea Tagliasacchi and Mark Pauly.

---

## 1 Introduction

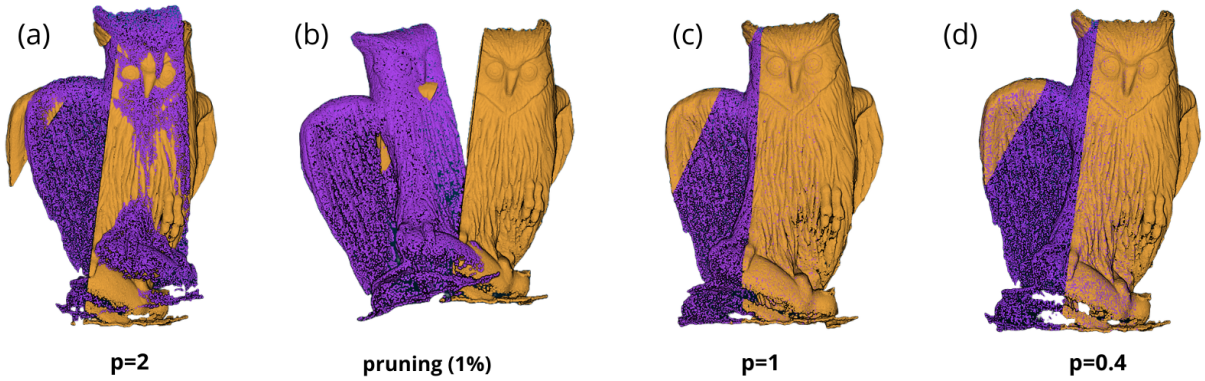


**Figure 1:** *Solving for optimal rigid alignment for incomplete geometry. (a) Traditional least-squares ICP does not distinguish between inliers and outliers, resulting in poor alignment. (b) The "correspondence pruning" ICP, a method to solve the problems encountered by the classical ICP does not succeed. (c,d) The SICP or the '$l_p$-ICP', with $p \in [0, 1]$ robustly handle large amounts of noise, outliers and align perfectly the two clouds (especially for $p << 1$).*

To deal with situations as surface reconstruction or shape matching, we usually use the *Iterative Closest Point Algorithm* (ICP). This algorithm partially solves the problem of matching two surfaces which are not perfectly aligned but in coarse alignment.

It works by alternating between 2 steps :

- A closest point computation (correspondence between 2 datasets)
- Solving the optimal rigid transformation (to go from correspondence to alignment)

The main issue with this method is the sensitivity to outliers and missing data. Such problems arise in most applications. For instance when aiming at reconstructing a 3D surface from many scans one will always get some occlusions due to different points of view (missing data) or with devices at consumer level which make a lot of errors as Kinect (outliers).

Several variants of ICP try to tackle these issues. Some implementations suppress some "unreliable" correspondences (by introducing a threshold). Other implementation re-adjust the weights of correspondences (putting low weight on "unreliable" correspondences). Nevertheless most of those adjustments require manual assistance. *Sparse Iterative Closest Point* (SICP) introduces a sparsity inducing norm which will address these issues without the need of any external intervention while keeping the simple ICP structure.

## 2 Theoretical Content

**Re-writing of the problem.**
The quality of the alignment (2nd Step) in ICP is related to the quality of the correspondences (1st Step) and incorrect closest point is very common. To tackle this issue, we want somehow to reduce or delete the correspondences which have a big distance at the previous iteration. This lead us to re-interpret the ICP problem as follow, with an $l_p$ norm with $p \in [0, 1]$:

$$(1) \qquad \arg\min_{\mathbf{Y}} \sum_{i=1}^{n} \|Rx_i + t - y_i\|_2^p + I_{\boldsymbol{Y}}(y_i)$$

$$(2) \qquad \arg\min_{\mathbf{R,t}} \sum_{i=1}^{n} \|Rx_i + t - y_i\|_2^p + I_{SO(k)}(R)$$

The two steps of the classical ICP can clearly be identified but an exponent $p$ is introduced to induce sparsity into the model. The idea is that it will grant a lower weight to an outlier. Indeed, if the residual error $Rx_i + t - y_i$ is large due to an outlier, the model will give less importance to this error if p is small. So Minimizing this will gives us a transformation less sensitive to aberrant data.

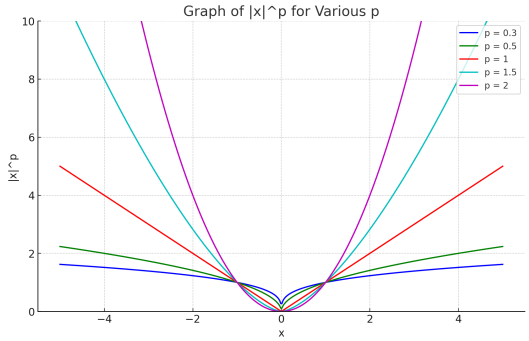


**Figure 2:** *Penalty functions used to induce sparsity in the optimization.*

Figure 2 illustrates the intuition at the origin of SICP. Indeed as p decreases, the curve becomes flatter indicating a lower sensitivity to changes in x. In our context, this means that the influence of an outlier decreases with p, by reducing the weight of larger errors.

**Non smooth problem.** Adding the power p leads to a more robust model but the issue is how to the optimization problems that becomes non smooth and non convex. The problem is non smooth due to the absolute value and non convex due to the $p$ exponent (which can be solved by adding a square in the classical ICP).This implies that we do not get a simple close form as in the classical ICP, we will have to use "smart" optimization algorithms to find a solution.

**1st Step : Correspondences.** For the resolution of (1): we observe that since $\phi(r) = |r|^p$ is non decreasing on $\mathbb{R}^+$, and since $||.||_2$ is positive, the function $||.||_2^p$ achieve its minimum value at the same point as $||.||_2$. Thus we can use the same techniques as in the classical ICP for the correspondences. Typically a *Kd-tree* is constructed to find efficiently the neighbors of each point.

**2nd Step: Alignment.** A first step in the resolution of (2) consist in approximating this problem by a more simple one. [CY08] suggests to solve the following weighted least squares problem :

$$\arg\min_{\mathbf{R,t}} \sum_{i=1}^{n} w_i^{p-2} \|Rx_i + t - y_i\|_2^2 + I_{SO(k)}(R)$$

where $w_i$ is the $l_2$ residuals from the previous iteration.
It clearly make some sense to achieve what we want, but when $w_i$ is very small (≈no error) then $w_i^{p-2}$ (with $p \leq 1$) is going to tend to $\infty$, which can cause numerical problems.
However we can directly solve the non convex problem with an optimization technique : *the Augmented Lagragian Method* (ALM). To do so we rewrite (2) as:

$$\arg\min_{\mathbf{R,t,Z}} \sum_{i=1}^{n} \|z_i\|_2^p + I_{SO(k)}(R) \;\; \textbf{s.t} \;\; \delta_i = 0$$

where $\delta_i = Rx_i + t - y_i - z_i$. This gives us the following ALM function:

$$\mathcal{L}(R, t, Z, \Gamma) = \sum_{i=1}^{n} \|z_i\|_2^p + \lambda_i^T \delta_i + \frac{\mu}{2} \|\delta_i\|_2^2 + I_{SO(k)}(R)$$

where $\lambda_i$ Lagrange multiplier and $\mu$ a penalty weight. It is important to note (for parameters tuning) that the larger $\mu$ is, the more the constrained equality is respected.
This optimization problem can be solved by using the *Alternating Direction Method of Multipliers* (ADMM), a non convex optimization algorithm which can be interpreted as an (or more than one) iteration of block coordinate descent on the ALM function. It decomposes the problem into the three following steps:

$$\textbf{Step 2.1:} \quad \arg\min_{Z} \sum_{i=1}^{n} \|z_i\|_2^p + \frac{\mu}{2} \|z_i - h_i\|_2^2$$

$$\textbf{Step 2.2:} \quad \arg\min_{\mathbf{R,t}} \sum_{i=1}^{n} \|Rx_i + t - c_i\|_2^2 + I_{SO(k)}(R)$$

**Step 2.3:** $\lambda_i = \lambda_i + \mu\delta_i$

where $c_i = y_i + z_i - \frac{\lambda_i}{\mu}$ and $h_i = Rx_i + t - y_i + \frac{\lambda_i}{\mu}$.
It comes easily that *step 2.1* is equivalent to:

$$z_i^* = \begin{cases} 0 & \text{if } \|h_i\|_2 \leq \tilde{h_i}, \\ \beta h_i & \text{otherwise} \end{cases}$$

where $\tilde{h_i}$ and $\beta$ are known values.
It is important to note that step 2 is the same as in the classical algorithm but with modified points cloud. So we have the full algorithm, we just have to keep in mind that we need to iterate *steps 2.1,2.2,2.3* to get the corresponding transformation R,t and then proceed as in the ICP algorithm. To sum up there are (at least) two loops in this algorithm.

**Point To Plane.** Let us emphasize that the classical ICP is formulated as a point-to-point optimization, but it exists point-to-plane variant. It permits to simplify our optimization problem and to obtain a better convergence speed.
The principle is that projection of $x$ onto the closest point of the reference cloud $y$ can be approximated by $n^T(x-y)$, with $n$ the normal at point $y$. For correspondences we can proceed as in the previous part because we are only searching for nearest point. And only then we are going to introduce the normals information with the alignment step. This second step is found by applying the same reasoning that for the point-to-point version. We get the following three ADMM steps :

**Step 2.1:** $\arg\min_Z \sum_{i=1}^{n} |z_i|^p + \frac{\mu}{2}|z_i - h_i|^2$

**Step 2.2:** $\arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{n} |\delta_i - c_i|^2 + I_{SO(k)}(R)$

**Step 2.3:** $\lambda_i = \lambda_i + \mu\delta_i$

where $\delta_i = n_i^T(Rx_i + t - y_i)$, $c_i = z_i - \frac{\lambda_i}{\mu}$ and $h_i = \delta_i + \frac{\lambda_i}{\mu}$.
*Step 2.2* is a simple least square point-to-plane optimization. The important point is that the optimization in *Step 2.1* is made on a scalar $z_i$. The point-to-plane version gets rid of the dimension, and so we expect the optimization to be much faster. The optimization is made in the same fashion as in the point-to-point version, with the shrinking of $z_i$.

**Summary and profit.** To resume, the SICP takes advantage of the property of the $.^p$ function to reduce the penalty on the outliers. It creates a kind of re-weighting without actually creating one, which avoids having to face numerical issues. The optimization problem is solved through ALM techniques which permits to deal with a non convex objective.
The benefit that will gain this method on the classical ICP is that it will not try to perfectly align the two clouds at all cost if they are not supposed to align. In numerous real-life examples, one aims at minimizing the distance between point clouds in cases where there is a partial match or overlap between them which makes the SCIP more relevnat than SICP. Indeed the ICP will be tempted to align completely the two clouds (due to the least square minimization). These are nice theoretical properties of the SICP illustrating its superiority to others sparsing techniques, that need to be tested in practice.

# 3 Implementation

**Datasets.**
Most experiments were conducted on the same datasets as in the paper (*owl.ply*, *monkey.ply*, *bears.ply*). These datasets can be found at : https://lgg.epfl.ch/statues.php.. 1ll experiments were first tested on the classical *Standford Bunny* because it's a small and convenient dataset for our experiments. For some experiments, we made the following transformations of the datasets to improve the set up to test the algorithms:

- *segmentation* : We carefully crop the clouds into two separate clouds (*ref* and *data*) taking into account the existence of a consequent overlapping between them. It seems a good way to test SICP, as we can check if it can bring closer only the overlapping part and not the full cloud (contrary to ICP). Most of this segmentation is made with *CloudCompare*.

- *transformation* : We apply some linear transform to one of the two clouds. The aim is to check if our algorithms are capable of bringing the clouds closer even if they are transformed (returned, far away, slightly shrink ...), which often occurs in practice.

- *noise* : We add a small Gaussian noise on both clouds as it will better correspond to real-life examples (many imprecision on measurements in practice).

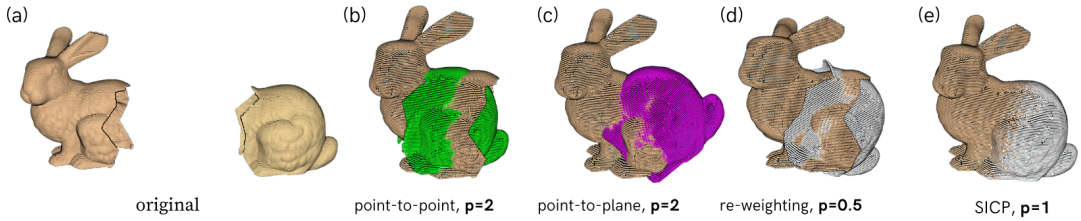


**Figure 3:** *(a) Visualization of the alignment of the Standford Bunny tested with different algorithms. (b,c) Traditional ICP with the point-to-point and point-to-plane version. (d) Traditional ICP is combined with re-weighting. The convergence is difficult due to numerical problems. (e) Sparse ICP (with p=1) outperforms the other models*

- *aberrant data* : We add aberrant data points to *ref* cloud. To do so we add points at a Gaussian distance from a random point in *ref* cloud. It could permits to play the role of other objects far in the scene that the ICP could misclassify as a point belonging to the *ref* cloud. And then see if the SICP could handle those aberrant point.
- *decimate* : We discard a given percentage of points in both clouds. Some of the datasets are really too large (more than $10^6$ points) and reducing them, even by 80% for example, will not alter the spirit of the cloud.

**Algorithms.**
The algorithms are implemented in Python, unlike those in the initial paper that were implemented in C++ and are available here. C++ has many advantages and is better suited for this kind of algorithm, especially for the rapidity of loops execution, but the implementation in Python can be more broadly understood and applied. For now, this implementation is quite slow compared to the original one. But it could be improved by parrallelizing loops of some algorithms. In this repository, we provide the implementation of:

*Traditional ICP* : We implemented both of the point-to-point and point-to-plane algorithm. For the point to plane one, the implementation of the *best_rigid_transform* function relies on [Low04].

*ICP with correspondences pruning*: It consists of the Traditional ICP registration combined with correspondence pruning, where correspondences with a distance above $\delta$% of the diagonal bounding box are rejected. We only compute the second step using "accepted" point. We implemented the point-to-point version.

*Re-weight ICP*: The problem formulation of this algorithm is given by the equation *reweight*. We could solve it in the exact same way as in the traditional ICP replacing the re-centered cloud $q_i = p_i - \frac{1}{n}\sum_{i=1}^{n} p_i$ by $q_i = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i p_i$. Based on [CY08], we implemented the point-to-point version.

*Sparse ICP* : We implemented the point-to-point and the point-to-plane version. We heavily relied on the insights from the paper [BTP13] and extensively utilized their repository.

# 4 Experiments

**Reproducing results.**
We first wanted to check which variant of SICP was the fastest to converge. It is very important to use the most efficient one, as a single iteration of SICP in Python could last more than 30 seconds (for a $3 * 10^4$ points 3D cloud).

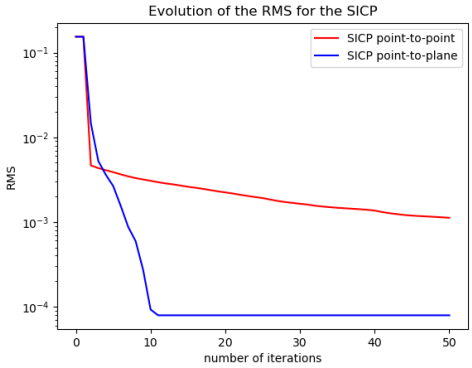


**Figure 4:** *Convergence of the point-to-point and point-to-plane SICP. The point-to-plane outperform the point-to-point version. In all of our experiments we use the point-to-plane SICP.*

We also wanted to reproduce the experiment made in the paper. We provide two of these results with **Figure 2** and **Figure 3**

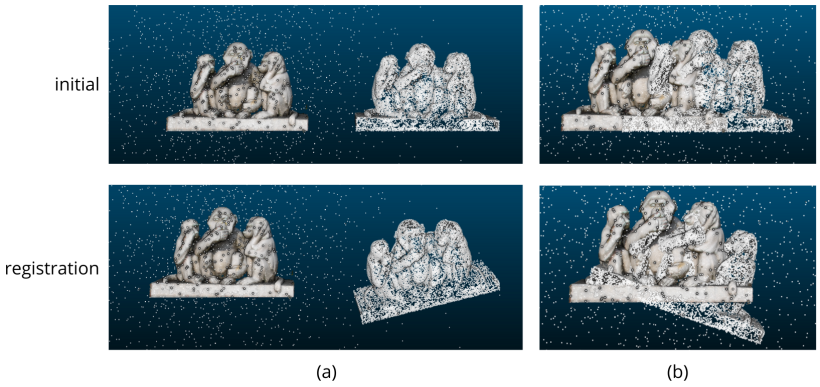


**Figure 5:** *We align two models with partial overlap and a large number of outliers. As SICP employs closest point correspondences, we do not converge to the correct solution (a). Even when source and target are in relatively close proximity, the alignment stays of poor quality (b).*

We also conducted an additional experiment, as represented in **Figure 5**. It illustrates the problem of the first step correspondences. Note that a different result is obtained in the original paper.

**Future work.** We have not designed any stopping criteria for the SICP. It is an important feature of this kind of algorithm, as the iteration could take considerable time. We could use the duality conditions as stopping criteria as the ALM techniques provides one or simply stop the algorithm when the RMS is too close from one iteration to another.

We should make more tests about first correspondences step with a large numbers of outliers and with other datasets.

We have not implemented the point-to-plane version of *Re-weight ICP* and *ICP with correspondences pruning*. It seems necessary, to fairly compare methods and accelerate the process.

# 5 Limits

**Execution time.** A major issue of this algorithm is the time of execution. We know that one of the principal challenges is to be able to registrate really fast. For example, SICP may not be suitable for real-time applications such as autonomous driving. This is due to its computational intensity, which is not compatible with the rapid decision-making required nowadays. Therefore, faster, though perhaps less robust, alignment algorithms are generally preferred in such scenarios, such as registration algorithm involving deep learning.

**Selection of p.**
Selecting the appropriate value for p presents a difficulty. On the one hand, smaller values of p have been shown to yield better alignment results. However, the trade-off is that the smaller p becomes, the more computationally intensive the algorithm gets, leading to longer convergence times. This complexity means that finding the optimal p value requires balancing the need for high-quality alignments against the practical constraints of processing time.

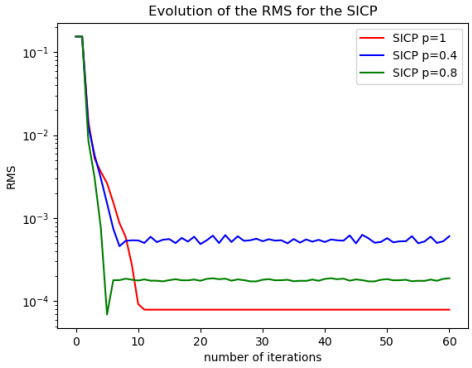


**Figure 6:** *Convergence of SICP for various p. The $l_1$-ICP converging much faster than other $l_p$-ICP with $p < 1$.*

Moreover, with this theoretical analysis, the choice of p is limited in $]0, 1]$ due to the way *Step 2.1* is solved. It would be interesting to be able to use the $l_p$-ICP for $p \in [1, 2]$ because of the trade-off between speed of convergence and robustness. This choice wouldl provide a more robust version than ICP because $p < 2$ and a faster version than SCIP because $p > 1$.

**Hyper-parameters tuning.** The choice of the parameter $\mu$, which is the penalty weight, is not explained in the original paper. This parameter plays a critical role in the SICP algorithm as it directly influences the balance between fitting

the data and enforcing sparsity in the solution. Even if it is possible to gradually increase its value through the algorithm, we need to fix an upper bound. Its optimal value can significantly vary depending on the specificity of the cloud, such as the level of noise or the presence of outliers. Finding the right $\mu$ value thus requires careful tuning, often through trial and error or parameter search techniques, making it a challenging aspect of applying SICP effectively to diverse datasets.

**Correspondences.** The goal of the SICP is to imitate the traditional ICP while avoiding to be fooled by wrong correspondences. As we have seen, the SICP only intervenes in the second step (alignment) to reduce the effect of wrong correspondences. But instead of limiting this first effect, we could directly suppress it. For now, for all algorithms we make the correspondences for an $x \in data$ by taking the first closest point $y \in ref$ but $y$ could be an outlier. **Figure 5** illustrates why this first step could be problematic.

A possible solution would be to directly act on this first step, by only taking into account correspondences for which the next 2 to k closest points in *ref* of x are at a "reasonable" distance from $y$. We could also average the k nearest points in *ref* of x. Even if these manipulations could take some extra time, they could make the optimization easier. We could directly proceed to the second step of the ICP because the sparsity is already taken care in the first step.

Moreover, it will not be necessary to tune a parameter as in a correspondences pruning way.

# 6 Conclusion

The SICP paper provides a robust extension of the classical ICP that addresses the problem of outliers during the registration with strong theoretical guarantee. The practical experiments seem to validate its efficiency and robustness. Despite its advantages, it remain uncertain if SICP will become the state-of-the-art algorithm due to its execution time and correspondence issues.

# References


[BTP13] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In Yaron Lipman and Richard Hao Zhang, editors, *Eurographics Symposium on Geometry Processing*, volume 32. École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2013.

[CY08] Rick Chartrand and Wotao Yin. Iteratively reweighted algorithms for compressive sensing. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on.* IEEE, 2008. Source: IEEE Xplore.

[Low04] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Department of Computer Science, University of North Carolina*, 2004.